MASTER EN SCIENCES INFORMATIQUES

Conception et réalisation d'un outil de consultation de lexiques multilingues en traduction automatique

Thunus, Véronique

Award date:
1997

Facultés Universitaires Notre-Dame de la Paix de Namur
Institut d'informatique

CONCEPTION ET REALISATION D'UN OUTIL DE CONSULTATION DE LEXIQUES MULTILINGUES EN TRADUCTION AUTOMATIQUE

Mémoire réalisé par Véronique THUNUS

en vue de l'obtention du diplôme de « Maître en Informatique »,

suite à son stage en entreprise au Centre de Recherche Public Centre Universitaire
162a avenue de la Faïencerie, 1511 Luxembourg, Luxembourg,
du 1er septembre 1996 au 31 décembre 1997.

Promoteur : Jacques Berleur s. j.
Co-promoteurs : Guy Deville
Pierre Mousel

Année académique 1996-1997

SOMMAIRE

Le but de ce travail est de présenter le projet européen Apollo qui a pour objectif la création et la maintenance de documents multilingues dans les secteurs bancaire et financier. Ce projet est en fait une maquette précédant le développement d'un banc de travail complet et a une durée de 15 mois. Il a rassemblé dix partenaires dans cinq pays européens différents.

On analysera chaque composant d'Apollo tel que le traitement de textes dans lequel il s'inscrit, le traducteur automatique ou le module de gestion de versions de documents multilingues et on les enchaînera les uns aux autres en présentant l'architecture de ce projet.

Dans ce mémoire, on s'intéressera particulièrement à la partie d'Apollo concernant le dictionnaire multilingue, qui a fait l'objet de mon stage effectué de septembre 1996 à janvier 1997 au Centre de Recherche Public - Centre Universitaire à Luxembourg.

Ainsi on étudiera l'ensemble des composantes de ce dictionnaire : son lexique, son interface et les bases de données auxquelles il accède. Ensuite, on explicitera son fonctionnement et son implémentation.

ABSTRACT

The purpose of this work is to present the european project Apollo, which aims to create and maintain multilingual documents in the sectors of banking and finance. This project actually is a mockup that preceed the development of a complete workbench and has a duration of 15 months. It brought ten partners of five different european countries together.

We will analyse each component of Apollo, such as the word processing, within it works, the machine translator or the multilingual documents versionning module and we will assemble these components by presenting the architecture of this project.

In this thesis, we will put our interest more particularly on the multilingual dictionary of Apollo, on which I have been working during my training from september 1996 to january 1997 at Centre de Recherche Public - Centre Universitaire in Luxembourg.

We will study all components of this dictionary : its lexicon, its interface, its databases. Then, we will explain its working processus and its implementation.

REMERCIEMENTS

Je remercie particulièrement mon promoteur le Père Jacques Berleur ainsi que mes co-promoteurs Monsieur Guy Deville et Monsieur Pierre Mousel pour l'aide précieuse qu'ils m'ont fournie lors de mon stage ou lors de la rédaction de ce mémoire.

Je remercie également toute l'équipe du Centre de Recherche Public - Centre Universitaire pour l'accueil chaleureux qu'elle m'a réservé et pour tout le temps qu'elle m'a consacré.

Table des matières

Glossaire

- **API** : Application Programming Interface : Ensemble de routines facilitant la création d'applications. L'API Windows par exemple, offre des fonctions et des types de données pour la création d'une application Windows.

- **Concept** : Représentation intellectuelle d'un objet conçu par l'esprit [LAROUSSE]. Dans les fichiers Spectrum et ABB utilisés lors de la création du lexique, à un concept ou à un mot dans l'interlangue correspond une entrée multilingue de 100 champs.

- **Corpus** : Ensemble de textes, de documents fournis par une tradition ou rassemblés pour une étude, en particulier pour une étude linguistique [LAROUSSE]. Le corpus APOLLO représente donc l'ensemble des textes qui nous ont été fournis par les différents partenaires du projet.

- **DTD** : Document type definition : Ensemble de règles définissant un document. Le DTD APOLLO définit les règles qu'il faut respecter pour avoir un document APOLLO.

- **Interlangue** : Langue de référence pour identifier un concept. Dans APOLLO, c'est le néerlandais qui a été choisie comme interlangue et c'est donc le mot néerlandais qui identifie un concept.

- **Lexique** : Ensemble des mots formant la langue d'une communauté et considéré abstraitement comme l'un des élément constituant le code de cette langue (notamm. par opp. à la grammaire) [LAROUSSE]. Il s'agit en fait d'un vocabulaire utilisé dans un domaine ou dans une langue.

- **Morphème** : Unité minimale de signification. On distingue les morphèmes grammaticaux (par ex. -ent, marque de la troisième personne du pluriel des verbes) et les morphèmes lexicaux (par exemple voi- dans voient) [LAROUSSE].

- **SGML** : Standard Generalized Marking Language. Ce langage décrit la structure de n'importe quel document sous forme standard. Il décrit aussi bien la forme du document que son contenu.

- **Système de synthèse** : Système informatique traitant le langage naturel et responsable de la production de résumés de textes avec ou sans l'intervention de l'homme.

- **Traduction automatique** : Système informatique responsable de la production de traductions d'une langue naturelle vers une autre, avec ou sans l'intervention de l'homme. Ce sont des systèmes où les traducteurs et les utilisateurs aident l'ordinateur à produire une traduction. L'homme peut intervenir lors de la préparation des textes à traduire, lors du processus de traduction ou lors de la révision des textes traduits [SABAH].

Introduction

Comme le souligne Jacques Berleur [BERLEUR], les premiers travaux informatiques traitant le langage naturel ont été réalisés peu après la naissance des ordinateurs. Aujourd'hui, de plus en plus d'applications informatiques s'y intéressent et on retrouve notamment beaucoup systèmes ayant pour objectif à la traduction automatique de textes, tel que le système Systran [HEYMANS] développé dès 1968 et qui traduit les différentes langues européennes. Ce système offre ses services dans le World Wide Web à l'adresse suivante : « http://www.systranmt.com »

Le projet Apollo se consacre également à la traduction automatique. En effet il a pour but de fournir à l'utilisateur un banc de travail pour la création et la maintenance de documents multilingues dans le secteur bancaire et financier.
Apollo est un projet européen pour lequel j'ai réalisé un stage de quatre mois au Centre de Recherche Public - Centre Universitaire à Luxembourg. Mon travail a consisté en l'implémentation d'un dictionnaire multilingue qui pourrait être une aide à la traduction pour un utilisateur d'Apollo.

Le but de ce mémoire est d'expliquer le projet Apollo lui-même ainsi que le développement du dictionnaire multilingue.

Tout traitement du langage naturel fait appel à des concepts et méthodes issus de la linguistique. Le premier chapitre de ce mémoire se consacrera donc au lexique qui a été élaboré pour plusieurs composants du projet Apollo et plus précisément pour le dictionnaire multilingue.
Dans ce chapitre, on donnera d'abord une classification des dictionnaires électroniques en distinguant les dictionnaires conçus pour les personnes de ceux qui sont intégrés dans des systèmes et qui ne sont alors accessibles qu'aux programmes. Ensuite, on mentionnera les données qui devraient figurer dans ces différents dictionnaires. Cette classification provient de Roy J. Byrd [WALKER]. Enfin, on comparera cette classification et ces données au dictionnaire multilingue construit pour le projet Apollo et on étudiera en détail le lexique qui a été utilisé dans le projet Apollo.

Le second chapitre étudiera la façon dont on a pu rassembler ces ressources lexicales utilisées par le dictionnaire multilingue. En effet, il a fallu trouver des données lexicographiques. Celles-ci ont alors été retravaillées pour former le lexique décrit au chapitre 1.

Le chapitre 2 se base également sur de la théorie proposée par Gunnel Engwall [ATKINS]. Cette théorie nous apprendra d'abord les différents types de textes que l'on peut rassembler pour former un corpus. En effet, on peut construire des corpus à partir de textes littéraires ou de journaux parlés,... Ensuite, elle nous précisera les caractéristiques de ces textes. Ici, on distinguera par exemple les sources orales des sources écrites. Sur cette base, j'essayerai de qualifier les textes du corpus Apollo.
C'est également dans ce chapitre qu'on verra quels sont les usagers de ce corpus.

Avant d'expliquer la conception du dictionnaire multilingue, il paraît utile d'en donner son cadre opératoire. Dans le troisième chapitre, on examinera donc le projet Apollo de façon détaillée.
D'abord, on étudiera les différentes tâches que requiert ce projet. Elles vont de l'implémentation pure et simple à l'information des utilisateurs.
Ensuite, on analysera toutes les composantes du projet. On essayera de comprendre le traitement de textes Interscript, le module de traduction automatique, le module de gestion de versions des documents multilingues et enfin le dictionnaire multilingue.
C'est donc dans ce chapitre que l'on expliquera l'architecture technique du projet Apollo. Au travers de cette architecture, on verra l'intégration des différents modules.

Le dernier chapitre sera consacré au développement du dictionnaire multilingue. On verra tout d'abord les fonctionnalités et l'objectif qu'on donne à ce dictionnaire. Ensuite on étudiera les différents éléments qui composent ce dictionnaire tel que l'interface et les bases de données. En effet, on verra comment on utilise l'interface du dictionnaire qui est intégrée dans le traitement de textes Interscript. On explicitera également comment les bases de données ont été remplies et rendues accessibles. Ensuite on étudiera comment le dictionnaire multilingue accède aux données contenues dans la base et comment il affiche l'information demandée par l'utilisateur dans l'interface. Ceci nous aidera donc à comprendre le mode de fonctionnement général du dictionnaire.

Chapitre 1. Cadre théorique : dictionnaires et lexiques

1. INTRODUCTION

La linguistique est définie par le petit Larousse illustré comme « la science qui a pour objet l'étude du langage et des langues. »
De plus en plus d'applications informatiques s'intéressent à cette science. Jacques Berleur [BERLEUR] identifie cinq classes de travaux informatiques portant sur le langage naturel : l'élaboration d'outils d'analyse linguistique, l'enseignement des langues assisté par ordinateur, la compréhension du langage naturel, la reconnaissance de la parole et la traduction automatique.
Ce mémoire s'intéresse plus particulièrement à la traduction automatique que [HUTCHINS] définit comme un système informatique responsable de produire des traductions d'un langage vers un autre avec ou sans l'intervention de l'homme.
Ce chapitre ne prétend pas décrire les différents systèmes de traduction automatique, ni la traduction automatique en général. Il sera essentiellement consacré à l'étude des dictionnaires de consultation, utilisés entre autre en traduction automatique.

Après avoir donné quelques généralités sur les lexiques, j'envisagerai les différents types de dictionnaires. Ensuite, j'étudierai les diverses entrées lexicales intégrées dans chaque type de dictionnaires. Finalement, je décrirai le contenu du lexique utilisé pour le dictionnaire multilingue d'Apollo.

2. LE LEXIQUE

La composante essentielle d'un dictionnaire de consultation est le lexique, c'est à dire l'ensemble des mots qui vont pouvoir être recherchés dans ce dictionnaire et l'information qui se rapporte à chaque mot.
Selon le type du dictionnaire et l'utilisation qu'on veut en faire, le contenu du lexique sera différent.
En général, une entrée lexicale d'un dictionnaire contient des informations orthographiques, relatives à l'écriture du mot, des informations phonétiques, relatives à la prononciation du mot,

des informations morphologiques, relatives à la forme des mots et exprimant les liens entre les différentes parties formant le mot, des informations syntaxiques comme la catégorie, expliquant la manière dont le mot peut se combiner à d'autres pour former des phrases ou expressions et des informations sémantiques relatives aux sens du mot. Ces dernières informations donnent une définition et des exemples de l'utilisation du mot dans un certain sens et aident à trouver des synonymes et des antonymes à ce mot.
La plus grande différence entre les lexiques réside dans les relations entre les mots, les entrées lexicales et le sens des mots.

3. LES DIFFERENTS TYPES DE DICTIONNAIRES

Roy J. Byrd [WALKER] distingue deux types de dictionnaires électroniques : les dictionnaires pour les personnes et les dictionnaires pour les programmes.
Les dictionnaires pour les personnes sont accessibles à l'utilisateur par une interface à leur poste de travail. Cette interface permet à l'utilisateur de faire des requêtes concernant les mots contenus dans les dictionnaires. Ces dictionnaires peuvent être des applications à elles toutes seules, mais ils peuvent aussi faire partie d'autres applications comme le correcteur orthographique contenu dans un traitement de textes ou comme un dictionnaire terminologique faisant partie d'un système de traduction.
Les dictionnaires pour les programmes fournissent de l'information à laquelle l'utilisateur ne pourra accéder que de manière indirecte. En effet, les utilisateurs d'un système de synthèse ou de critique de textes ne verront jamais ces informations directement à l'interface. Ce sont plutôt les résultats de l'application utilisant le dictionnaire qui sera affiché à l'écran. Les requêtes proviennent donc des applications; ce sont elles qui accèdent aux dictionnaires.
Détaillons d'avantage ces deux types de dictionnaires.

3.1. Les dictionnaires pour les personnes

Ces dictionnaires fournissent directement de l'information à l'utilisateur en l'affichant à l'écran.
A nouveau, Roy J. Byrd [WALKER] fait une distinction entre les dictionnaires en ligne et ceux intégrés dans un système.

3.1.1. Dictionnaires en ligne

L'utilisateur d'un tel système fournit un mot au dictionnaire. Le module de recherche d'informations accède à la base de données et fournit le résultat de al recherche au module d'affichage. Celui-ci affiche l'information qu'on possède à propos de ce mot dans une interface. En général, on retrouve dans ces dictionnaires la même information que celle qu'on trouve dans les dictionnaires papier.

Il importe de décider quelles informations vont être présentées en réponse à une requête. Ceci dépendra de l'information contenue dans le lexique. La section 4 traite de ces contenus pour les différents types de dictionnaires.

Etant donné la dimension de l'écran, toutes les informations concernant un mot pourront rarement être affichées en une fois. Une solution est de travailler par niveau. Si la requête porte par exemple sur un mot possédant plusieurs sens, le système répondra d'abord en donnant les différentes définitions de ce mot. Et lorsque l'utilisateur en aura sélectionné une, de plus amples informations concernant ce sens du mot seront données.

Le rôle d'un dictionnaire de consultation est d'extraire l'information demandée et de la restructurer pour la présenter à l'utilisateur. Il est donc important de trouver une organisation de l'information sur le disque de l'ordinateur, qui permet de réduire le temps de réponse face à une requête. Cette organisation, dépendant du type de dictionnaire que l'on désire implémenter et de la quantité d'informations disponibles, se fera à l'aide d'un base de données.
Il faut donc concevoir une base de données relationnelle car elle donne une organisation efficace de l'information et permet de répondre dans un délais raisonnable aux requêtes de l'utilisateur. Elle doit être la plus flexible possible et doit contenir le plus d'informations utiles possible. Elle pourrait même être multifonctionnelle, c'est à dire qu'elle servirait peut-être à d'autres applications que celle du dictionnaire.
Le temps d'accès aux données dépend du matériel avec lequel on travaille, des données et du résultat attendu de la requête. Plus le processeur est rapide et plus le temps de réponse est bref. Si on demande beaucoup d'informations, il faut accéder à plusieurs tables et le temps de réponse sera plus long.
De plus, il est important d'avoir une interface qui facilite la navigation entre les différentes données, et donc éviter de devoir ouvrir et fermer trop de boîtes de dialogue.

Pour alimenter les lexiques du dictionnaire, on aura tendance à combiner plusieurs sources d'informations. De gros problèmes surviennent alors : deux sources distinctes peuvent donner des sens différents du même mot et il faut donc les faire correspondre, ce qui n'est pas tâche

aisée. Notons de plus qu'un dictionnaire donne parfois plus de sens différents d'un mot qu'un autre. Puiser de l'information dans différentes sources pour en faire une seule base de données est donc déconseillé, à moins de posséder une technologie de représentation des données qui pemettent de comparer les différents sens que deux dictionnaires donnent à un même mot.

Les dictionnaires en ligne ont la capacité d'offrir d'autres fonctions que la simple recherche d'information face à un mot. Leurs lexiques pourraient servir à créer des outils éducatifs ou des jeux de mots. Voici quelques exercices trouvés dans le dictionnaire des compétences de LDAE et dans Thorndike and Barnhart 1968: 26 [WALKER].

« In the sentences below, the words in dark type all have more than one meaning. Look them up in the dictionary and decide which meaning correctly explains the use of the word in the sentences, and write the number of word in the blank.

1. She is very **able** teacher. __
2. I can't pay you. I have no money in my bank **account**. __
3. His essay was **poor**, so he got a low mark for it. __
4. John just won't listen to **reason** any more. __
5. I can't leave early. I'm up to my **neck** in work. __ »

« Look up each of the following special meanings. Write a sentence using each one. After the sentence write the entry word under which you found the meaning.

crocodile tears
follow in one's footsteps
bury the hatchet
catch one's eye
keep one's head
take heart »

L'avantage certain des dictionnaires en ligne est que leur maintenance est facilitée. Le dictionnaire peut facilement être mis à jour. Et ceci est très important pour les dictionnaires contenant des termes technologiques évoluant à grande vitesse. De nouveaux termes peuvent rapidement être insérés.

Un autre avantage est la possibilité de publier un dictionnaire sous forme papier mais aussi sous un autre média. La version en ligne offre des outils intelligents de recherche d'informations ainsi que de présentation de l'information.

3.1.2. Le système à dictionnaire intégré

Ces dictionnaires se situent entre les dictionnaires en ligne et les dictionnaires pour les programmes. En effet, ils sont intégrés dans des applications accessibles à l'utilisateur et les informations contenues dans ces dictionnaires sont affichées directement à l'écran. C'est le cas des dictionnaires utilisés dans les traitements de textes par le correcteur orthographique ou de ceux utilisés pour donner une aide à l'utilisateur dans les systèmes de traduction. Ainsi, le dictionnaire propose des alternatives de traduction pour un mot ainsi que d'autres mots associés au premier tels que des synonymes ou antonymes.

3.2. Les dictionnaires pour les programmes

Ces dictionnaires fournissent de l'information lexicale à des applications traitant le langage naturel. Ici, ce sont les programmes qui demandent et utilisent l'information et non l'utilisateur. Les programmes, tels qu'un traducteur automatique ou un système de synthèse, font subir des transformations à l'information fournie par les dictionnaires avant de donner le résultat du traitement à l'utilisateur.

Un concepteur de dictionnaires pour programmes doit

- déterminer et acquérir le plus d'informations et de mots possible et constituer ainsi un dictionnaire de base qui servira plusieurs applications.
- fournir des outils de recherche d'information spécifique à chaque application ainsi que des moyens d'enrichir le dictionnaire de base de nouveau mots.
- proposer une représentation de l'information adaptable à chaque application utilisatrice.

4. LES DIFFERENTES DONNEES LEXICALES

Selon Jonathan Slocum et Martha G. Morgan [WALKER], les exigences des hommes diffèrent de celles des programmes du point de vue du contenu des lexiques des dictionnaires.

4.1. Les dictionnaires pour les hommes

4.1.1. Les dictionnaires généraux

Voici les informations contenues généralement dans un dictionnaire monolingue utilisé par les hommes :

- la forme graphique et phonétique du mot
- l'étymologie
- les modalités d'usage : région géographique, niveau de diction (poétique, vulgaire,...), domaine d'utilisation (chimie, architecture,...), jargon.
- les variantes morphologiques : référence à la conjugaison, à la déclinaison
- l'information grammaticale : catégorie grammaticale (nom, verbe, adjectif,...), sous-catégorie grammaticale (verbe transitif, réflexif,...), genre grammatical
- le sens
- les relations à d'autres mots : synonymes, antonymes
- les idiomes et contextes : exemples

Les dictionnaires multilingues contiennent aussi ces informations à quelques exceptions près. L'étymologie des mots se s'y retrouve pas souvent. Par contre, ils possèdent une information spécifique : la traduction du mot dans une ou plusieurs autres langues. Dans ce type de dictionnaire, les exemples sont plus explicites et plus nombreux.
Remarquons que, comme le souligne Ulrich Heid [BOUILLON], « ... les informations lexicales reprises dans les dictionnaires bilingues doivent être disponibles dans des dictionnaires monolingues »

Voici l'exemple de l'entrée d'un dictionnaire général anglais pour le mot « start »

1start \'stärt\ *vb* [ME *sterten*; akin to MHGzert to stand up stiffly, move quickly] *vi* (14c) **1 a** : to move suddenly aviolently: SPRING ⟨~*ed* angrily to his feet⟩ **b** : to react with a sudden brief involary movement ⟨~*ed* when a shot rang out⟩ **2 a** : to issue with sudden force ⟨blo~*ing* from the wound⟩ **b** : to come into being, activity, or operation ⟨when does movie ~⟩ ⟨the rain ~*ed* up again⟩ **3** : to protrude or seem to protrude ⟨eyes ~*ing* in their sockets⟩ **4** : to become loosened or forced out of place ⟨one of the planks ~*ed*⟩ **5 a** : to begin a course or journey ⟨~*ed* toward the door⟩ ⟨just ~*ing* out⟩ **b** range from a specified initial point ⟨the rates ~ at $10⟩ **6** : to begin an activir undertaking; *esp*: to begin work **7** : to be a participant in a game or contest; *esp*: ie in the starting lineup ~ *vt* **1** : to cause to leave a place of concealment: FIRH ⟨~ a rabbit⟩ **2** : *archaic*: STARTLE, ALARM **3** : to bring up for considerationdiscussion **4** : to bring into being ⟨~ a rumor⟩ **5** : to cause to become loosened ásplaced **6** : to begin the use of ⟨~ a fresh loaf of bread⟩ **7 a** : to cause to movet, or operate ⟨~ the motor⟩ **b** : to cause to enter a game or contest; *esp*: to put in tharting lineup **c** : to care for or train during the early stages of growth and devpment ⟨~*ed* plants⟩ ⟨a well-*started* coonhound⟩ **8** : to do or experience the fstages or actions of ⟨~*ed* studying music at the age of five⟩ **syn** see BEGIN

Source : Merriam-Webster's Collegiate Dictionary, 10th edn.

L'exemple suivant est l'entrée d'un dictionnaire général allemand pour le mot « starten »

'star|ten 1 ⟨V. i.⟩ *zu einem Rennen, Wettkampf ablaufen, abfahren, abspringen, abschwimmen; abfliegen, zum Flug aufsteigen* (Flugzeug); eine Rede ~ ⟨fig.; umg.⟩ *mit einer Rede beginnen;* ⟨umg.⟩ *abreisen* (bes. mit dem Auto) 2 ⟨V. t.⟩ *starten lassen* (Flugzeug, Rakete); eine Veranstaltung ~ ⟨fig.; umg.⟩ *eine V. stattfinden lassen;* ein Flugzeug wird mittels Schleuder gestartet [zu *Start*] **~ter** ⟨m. 3⟩ *jmd., der zum Rennbeginn das Zeichen gibt;* ⟨†⟩ = *Anlasser* (an Kraftfahrzeugen)

Source : Wahrig, G. (ed.), Deutsches Wörterbuch

Voyons maintenant l'exemple de deux entrées bilingues de dictionnaires généraux pour le mot « start(en) »

start[2]

2 *vt* (a) *(begin)* anfangen mit; *argument, career, new life, negotiations* beginnen, anfangen; *new job, journey* antreten. **to ~ work** anfangen zu arbeiten; **he ~ed life as a miner** er hat/hatte als Bergmann angefangen; **don't ~ that again!** fang nicht schon wieder (damit) an!; **to ~ smoking** das *or* mit dem Rauchen anfangen; **he ~ed coming late** er fing an, zu spät zu kommen.

(b) *(runners)* starten zu; *(cause to begin) runners, race* starten; *train* abfahren lassen; *rumour* im Umlauf setzen; *conversation* anfangen, anknüpfen; *fight* anfangen; *blaze, collapse, chain reaction* auslösen; *coal fire etc* anzünden; *(arsonist)* legen; *(found) enterprise, newspaper* gründen, starten *(inf)*. **to ~ sb thinking/on a subject** jdn nachdenklich machen/jdn auf ein Thema bringen; **to ~ sb in business/on a career** jdn zu einem Start im Geschäftsleben/zu einer Karriere verhelfen; **the discovery ~ed a new line of research** mit der Entdeckung kam eine neue Forschungsrichtung in Gang; **I don't want to ~ anything but ...** ich will keinen Streit anfangen, aber ...; **just to ~ you getting used to it** nur damit Sie sich erst mal daran gewöhnen; **as soon as she ~ed the baby** *(inf)* sobald sich das Baby angekündigt hatte; **when she wore the first miniskirt she didn't realize what she was ~ing** als sie den ersten Minirock trug, war ihr nicht bewußt, was sie damit auslösen würde; **look what you've ~ed now!** da hast du was Schönes angefangen *(inf)*.

(c) *car* starten; *engine also* anlassen; *clock* in Gang setzen; *machine, motor also* anwerfen.

(d) **to ~ a horse in a race** eine Nennung für ein Pferd abgeben.

3 *vi (begin)* anfangen, beginnen; *(car, engine)* anspringen, starten; *(plane)* starten; *(move off)* anfahren; *(bus, train)* abfahren; *(boat)* ablegen; *(rumour)* in Umlauf kommen; *(violins, cellos etc)* einsetzen. **~ing from Tuesday** ab Dienstag; **to ~ for home** (nach Hause) aufbrechen, sich auf den Heimweg machen; **to ~ for work** zur Arbeit gehen/fahren; **to ~ for London** nach London losfahren; **to ~ (off) with** *(adv) (firstly)* erstens, erst einmal; *(at the beginning)* zunächst; **what shall we have to ~ (off) with?** was nehmen wir als Vorspeise?; **I'd like soup to ~ (off) with** ich möchte erst mal eine Suppe; **to ~ after sb** jdn verfolgen; **to get ~ed** anfangen; *(on journey)* aufbrechen; **he finds it difficult to get ~ed in the morning** er kommt morgens nur schwer in Schwung *or* Gang; **to ~ on a task/journey/the food** sich an eine Aufgabe/auf eine Reise/ans Essen machen; **to ~ talking** *or* **to talk** zu sprechen beginnen *or* anfangen; **he ~ed by saying ...** er sagte zunächst ...; **don't you ~!** fang du nicht auch noch an!

Source : Terrel, P., Calderwood-Schnorr, V., Morris, W.V.A., and Breitsprecher, R. (eds.), Collins German-English English-German Dictionary

starten ['ʃtartən], **1.** *v.i. (Spt.)* start, *(coll.)* get away; *(Av.)* take off, take the air; **~ zu**, participate in, take part in, have entered for *(a contest)*; *(Spt.)* **zu früh ~**, make a false start, *(coll.)* jump the gun. **2.** *v.t. (Spt.)* start *(a race)*; *(fig.)* launch *(an undertaking)*, get *(s.th.)* started *or* going. **Starter,** *m. (Spt. Motor.)* starter. **Starterklappe,** *f. (Motor.)* choke.

Source : Betteridge, H. T. (ed.), Cassell's German-English English-German Dictionary, 1st Macmillan rev. edn.

4.1.2. Les dictionnaires spécialisés

Les dictionnaires papier spécialisés n'ont pas la possibilité d'évoluer au même rythme qu'avancent les termes techniques. D'où l'intérêt d'en faire des dictionnaires électroniques, où comme on l'a déjà mentionné, les mises à jours sont facilitées. Comparons le contenu de ces dictionnaires avec les généraux examinés précédemment.

- pas d'indication de prononciation
- pas d'étymologie
- plus d'information sur les modalités d'usage pour indiquer la discipline dans laquelle le mot est utilisé
- pas de référence à la conjugaison ou à la déclinaison
- le genre grammatical et la catégorie sont occasionnellement indiqués
- moins de sens, mais il est spécifié par les définitions
- les synonymes et antonymes ne sont pas considérés comme indispensables.
- les idiomes et contextes ne figurent que dans les encyclopédies

Les dictionnaires spécialisés multilingues possèdent évidemment une information en plus : la traduction du mot d'une langue vers d'autres.

L'exemple suivant montre une entrée d'un dictionnaire technique anglais pour le mot « monitor »

monitor [COMPUT SCI] To supervise a program, and check that it is operating correctly during its execution, usually by means of a diagnostic routine. [ENG] 1. An instrument used to measure continuously or at intervals a condition that must be kept within prescribed limits, such as radioactivity at some point in a nuclear reactor, a variable quantity in an automatic process control system, the transmissions in a communication channel or bank, or the position of an aircraft in flight. 2. To use meters or special techniques to measure such a condition. 3. A person who watches a monitor. [MIN ENG] *See* hydraulic monitor. [VERT ZOO] Any of 27 carnivorous, voracious species of the reptilian family Varanidae characterized by a long, slender forked tongue and a dorsal covering of small, rounded scales containing pointed granules.

monitor board [COMMUN] A console at which a supervising telephone operator sits and from which she can intercept calls being handled by other operators.

monitor control dump [COMPUT SCI] A memory dump routinely carried out by the system once a program has been run.

monitor display [COMPUT SCI] The facility of stopping the central processing unit and displaying information of main storage and internal registers; after manual intervention, normal instruction execution can be initiated.

monitoring key [ELECTR] Key which, when operated, makes it possible for an attendant or operator to listen on a telephone circuit without appreciably impairing transmission on the circuit.

monitor mode *See* master mode.

monitor operating system [COMPUT SCI] The control of the routines which achieves efficient use of all the hardware components.

monitor printer [COMMUN] A teleprinter used in a technical control facility or communications center for checking incoming teletypwriter signals. [COMPUT SCI] Input-output device, capable of receiving coded signals from the computer, which automatically operates the keyboard to print a hard copy and, when desired, to punch paper tape.

monitor routine *See* executive routine.

monitor system *See* executive system.

Source : McGraw-Hill Dictionary of Scientific and Technical Terms, 3rd edn.

Voici à présent une entrée d'un dictionnaire technique anglais-allemand pour le mot « monitor »

monitor *v* [Dp] /überwachen ‖ *n* [OS] / Monitor, Monitorprogramm, Überwachungsprogramm ‖ ~ (s. graphics terminal) ‖ **high resolution** ~ [GrDp] / hochauflösender Grafikbildschirm, HR-Grafikmonitor ‖ **viewing** ~ [VDT] / Monitor (Bildschirm) ‖ ~ **area** [OS] / Monitorbereich ‖ ~ **card** [OS] Monitorsteuerkarte ‖ ~ **card deck** [OS] / Monitorkartenstapel ‖ ~ **control** [OS] / Monitorsteuerung ‖ ~ **control card** [OS]/ Monitorsteuerkarte ~ **control card set** [Sw] / Monitorsteuerkartensatz ‖ ~ **control statement** [OS] / Monitorsteueranweisung

monitored control (s. open-loop control) ‖ ~ **control system** [NC] / überwachtes Regelsystem

monitor file [OS] / Monitordatei ‖ ~ **information** [OS] / Monitorinformation

monitoring *n* [Dp] / Überwachung ‖ [Swit] / Mithören, Mitlesen, Mitschreiben ‖ **automatic** ~ **system** [PCC] / automatisches Regelsystem ‖ **observation and** ~ **service** [Swit] / Beobachtungs- und Überwachungsdienst ‖ ~ **channel** [Swit] / Überwachungskanal (für Störungsrückmeldung) ‖ ~ **circuit** (s. local record connection) ‖ ~ **counter** [Swit] / Überwachungszähler ‖ ~ **count register** [Swit] / Überwachungszählregister (UZR) ‖ ~ **feedback** [NC] / stabilisierende Rückführung, Kontrollrückführung ‖ ~ **function** [Swit] / Überwachungsfunktion ‖ ~ **loop** [Swit] / Überwachungsschleife ‖ ~ **period** [Sw, Swit] / Überwachungszeit ‖ ~ **point** [PCC, Meas] / Meßpunkt, Meßstelle ‖ ~ **print** (s. local record) ‖ ~ **printer** [IO, Tx] / Kontrollblattschreiber, Kontrollmaschine ‖ ~ **program** (s. monitor program) ‖ ~ **routine** [Sw] / Überwachungsroutine

monitor jack [Serv] / Prüflinke ‖ ~ **jack panel** [Serv] Prüflinkenfeld ‖ ~ **job sequence** [OS] / Monitorablauffolge ‖ ~ **job stream** [OS] / Monitorablauffolge ‖ ~ **parameter card** [OS] / Monitorsteuerkarte ‖ ~ **phase** [OS] / Monitorphase ‖ ~ **program** [OS] / Monitor, Monitorprogramm, Überwachungsprogramm ‖ ~ **run** [OS] / Monitorlauf ‖ ~ **session** [OS] / Monitorlauf ‖ ~ **statement** [OS] / Monitoranweisung ‖ ~ **step** [OS] / Monitorschritt

Source : Brinkmann, K.-H., and Tanke, E. (eds.), Data Systems and Communications Dictionary: German-English English-German, 4th edn., 1984

C'est une entrée d'un dictionnaire technique allemand-anglais pour le mot « überwachen » que décrit cet exemple

überwachen v [Dp] monitor, supervise ‖ [Dp] / to maintain a (continual, etc.) check on ‖ **zentrales ~ des Taktes** [Swit] / centralized clock monitoring

Überwacher *m* [OS] / trace program, tracer ‖ ~ (Programm zur Prüfung neu eingelesener Programme) [PCC, Sw] / checking program ‖ ~ (s. Datenerfassungsüberwacher)

überwachtes Regelsystem [NC] / monitored control system

Überwachung *f* [Dp] / supervision, monitoring

Überwachungs·anzeige (s. Prüfanzeigeeinrichtung) ‖ **~blattschreiber** *m* [Swit] / supervisor's teleprinter (ST)

Source : Brinkmann, K.-H., and Tanke, E. (eds.), Data Systems and Communications Dictionary: German-English English-German, 4th edn., 1984

4.2. Les dictionnaires pour les ordinateurs

4.2.1. Les dictionnaires généraux

Les ordinateurs ont d'avantage besoin d'informations lexicales qui serviront à l'analyse ou à la transformation des données de départ, que de définitions de mots.
Pour étudier le contenu de tels dictionnaires, examinons le système Metal [BOUILLON] [WALKER].
Metal est un système de traduction assistée par ordinateur élaboré à l'université du Texas dès 1961. Au départ, il ne traduit que de l'allemand vers l'anglais. Ensuite, Siemens-Nixdorf a développé les paires de langues français-néerlandais, néerlandais-français et français-anglais.

Chaque entrée du lexique est une forme canonique.

Pour une entrée verbale, voici quelques données que l'on trouve :

- la catégorie lexicale (VST)
- le radical (ALO)
- les possibilités de combinaison avec le radical (HEAD) (cet élément n'apparaît pas dans les exemples qui suivent)
- les arguments qui désigne la description quantitative et qualitative de la valence des verbes (nombres d'actants, le caractère facultatif de certains arguments,...) (AGRS)
- l'auxiliaire (AX)
- le passif (PV)
- le type transitif (TT)
- ...

Pour une entrée nominale, on trouve :

- la catégorie lexicale (NST)
- le radical (ALO)
- le type sémantique (TYN)
- la forme du complément (FC)
- le genre grammatical (GD)
- ...

Dans le dictionnaire bilingue, les entrées du lexique sont jumelles, une pour chaque langue, et le contexte d'utilisation est précisé.

Voici d'abord un exemple d'une entrée du dictionnaire de traduction allemande suivie d'une entrée du dictionnaire anglais pour les verbes « starten-start ». Ces dictionnaires sont utilisés par le système Metal.

```
('starten'                                    VST
  ALO       'start'
  ARGS      ((($SUBJ N1 N0 (ICP INF+ INF-)) OPT ($DOBJ N1 (TYPE
              P0)))
             ($SUBJ N1) OPT ($DOBJ N0 (ICP INF+))))
  AX        ('haben' 'sein')
  CL        (IMP-4 INF-EN PAI-2 PAS-2 PP-GEET PRI-3 PRS-1)
  PV        ('durch' 'von')
  TT        (I T)
  AUTHOR    'WIESNER'
  DATE      '18-Jul-89'
  SITE      'MUC')

'starten' VST → 'start' VST
  Pref      S.0.0.00
  Tag       (GV)
  XFMS      (XFR-CLS-MAP :SLCAT PP :SLCAN 'über' :TLCAT PP
              :TLCAN 'via')
            (XFR-CLS-MAP :SLCAT PP :SLCAN 'nach' :TLCAT PP :TLCAN
              'after')
《Smi MUC Wiesner 25-Aug-89》

('start'                                      VST
  ALO       'start'
  ARGS      ((($SUBJ N1) ($DOBJ N1 (TYPE P1)) ($OCOMP N0 (ICP ING)))
             (($SUBJ N1) OPT ($DOBJ N1 ('TYN SEM PRO POT CNC)
                                        N0 (ICP INF+ ING))
                             ($ADV TMP))
             (($SUBJ N1) ($POBJ N1 (PREP 'on' 'with')) OPT ($ADV TMP))
             (($SUBJ N1) ($ADV DIR LOC)))
  CL        (G-ING I-0 P-ED PA-ED PR-S)
  ON        CO
  PV        ('by')
  TT        (I T)
  AUTHOR    'WIESNER'
  DATE      '28-Oct-92'
  SITE      'LRC')
```

Voici maintenant ces mêmes types d'entrées mais pour les mots « Anfang-beginning ».

```
('Anfang'                                    NST
   ALO        'Anfang'
   CL         (S-S/ES)
   DR         (NP RD)
   GD         (M)
   KN         CNT
   LINK       G-SG
   SX         (N)
   TYN        (LOC)
   AUTHOR     'LESLEY'
   DATE       '31-Dec-84'
   SITE       'MUC')

('Anfang'                                    NST
   ALO        'Anfäng'
   CL         (P-E)
   DR         (NP RD)
   GD         (M)
   KN         CNT
   LINK       G-SG
   PLC        (NF)
   SX         (N)
   TYN        (LOC)
   AUTHOR     'LESLEY'
   DATE       '31-Dec-84'
   SITE       'MUC')

'Anfang' NST ⟶ 'beginning' NST
   Pref       S.0.0.00
   Tag        (GV)
《Sni MUC Whiffin 4-May-88》

('beginning'                                 NST
   ALO        'beginning'
   CL         (P-S S-01)
   DR         (RD)
   FC         (PP)
   KN         CNT
   MC         ('of')
   MORPH      VERBAL
   ON         CO
   SX         (N)
   TYN        (ABS LOC TMP)
   AUTHOR     'CHAPMAN'
   DATE       '15-Nov-91'
   SITE       'LRC')
```

4.2.2. Les dictionnaires spécialisés

Les différents types d'informations contenus de ces dictionnaires ne diffèrent pratiquement pas des dictionnaires généraux, mais l'information elle-même est plus spécifique. Elle sera souvent classée différemment. Ainsi, dans certains dictionnaires, le premier sens du mot anglais « papier » sera « document, article » plutôt que la substance elle-même.

5. LEXIQUE UTILISE PAR LE DICTIONNAIRE MULTILINGUE APOLLO

Le dictionnaire multilingue implémenté dans le cadre du projet Apollo est un dictionnaire pour les personnes. Il intégré dans un traitement de textes. Son lexique est spécialisé dans le domaine bancaire et financier.

Les bases du lexique utilisé pour ce dictionnaire multilingue ont été fournies par Spectrum, l'un des partenaires du projet spécialisé dans l'élaboration et la maintenance de ressources lexicographiques. Ce lexique contient en effet 2 000 concepts bancaires ou financiers, leurs traductions en néerlandais, anglais, français et allemand, quelques informations morphologiques et syntaxiques ainsi qu'une définition des concepts en néerlandais.
Le partenaire ABB a également fournit son dictionnaire néerlandais - français de terminologie bancaire.

La grande différence entre la ressource Spectrum et la ressource ABB est que Spectrum possède une base de données contenant tous les concepts qui sont donc déjà structurés. La ressource ABB correspond à la copie d'un dictionnaire papier et ne comprend pas de marques séparant les types d'informations.

On pourrait distinguer quatre phases lors de la réutilisation de lexiques existant :
- identifier les différents types d'informations disponibles
- vérifier et corriger les données pour qu'elles correspondent à la structure définie
- restructurer l'information pour qu'elle réponde à l'utilisation qu'on veut en faire
- ajouter, automatiquement ou manuellement, des informations.

Ces phases ont été poursuivies dans le projet Apollo.

Ces différentes ressources, Spectrum et ABB ont donc été retravaillées pour qu'elles soient homogènes. Le résultat de ces transformations est une série de fichiers ayant tous le même format.

Ces fichiers sont utilisés dans plusieurs composants du projet Apollo, notamment par le système de traduction automatique et par le dictionnaire multilingue.
C'est le néerlandais qui a été choisi comme interlangue car la seule définition que l'on possède dans le fichier Spectrum est en néerlandais. Cette définition donne le sens du mot et peut donc déterminer un concept. Dans les fichiers, le mot néerlandais est donc devenu l'identificateur d'un concept.

Je vais maintenant décrire le contenu du fichier résultant des modifications des ressources Spectrum et ensuite celui résultant des ressources ABB.

5.1. Le fichier Spectrum

Ce fichier compte 1 365 entrées multilingues. Chaque entrée comprend 100 champs contenant différentes informations. Chaque concept a au plus quatre traductions par langue.

5.1.1. Les champs de 001 à 020

aanbetaling
acception
Een deelbetaling die verschuldigd is bij het afsluiten van een contract of levering, waarna de overblijvende schuld in iin of meer termijnen wordt voldaan.
COUNT
S_ENTITY
006
007
008
009
010
011
012
013
014
015
016
017
018
019
020

Ces champs fournissent des informations indépendantes de la langue.

- 001 : ce champ contient le concept en néerlandais. Il est l'identificateur du concept, le néerlandais étant choisi comme interlangue.

- 002 : ce champ indique le énième sens analysé du concept. Si ce concept n'a qu'un seul sens, ce champ contient la valeur « acception ». Par contre, s'il a plusieurs sens, la première analyse de ce concept donnera la valeur « 1st acception » à ce champ, la seconde « 2nd acception », et ainsi de suite.
- 003 : ici, on trouve une définition du concept en néerlandais
- 004 : ce champ vaut « count » (valeur par défaut) ou « mass ». C'est une information sémantique utile au module de traduction automatique, et qui n'est pertinente que pour les substantifs.
- 005 : ce champ précise le type sémantique du concept : « animal, event, a_entity, building, region, instr, s_entity, human, body, vehicle, feast, solid, liquid, gas, plant, field, institution »
- 006 à 020 : ces champs ne sont pas encore utilisés.

5.1.2. Les champs de 021 à 040

down payment
sing
subst
be_vs_ae + comment + restriction
025
-
num
cat
be_vs_ae + comment + restriction
030
-
num
cat
be_vs_ae + comment + restriction
035
-
num
cat
be_vs_ae + comment + restriction
040

Ces champs concernent les informations relatives à la traduction anglaise.

- 021, 026, 031, 036 : ces champs contiennent au moins une traduction anglaise du concept. « - » signifie « pas de traduction ».
- 022, 027, 032, 037 : ces champs précisent le nombre de la traduction anglaise : « sing, plur, num (= non précisé) »
- 023, 028, 033, 038 : on retrouve dans ces champs la catégorie syntaxique du mot anglais : « adj, expr, p-expr (= expression commençant par une préposition), subst, verb, subst ; verb (= substantif ou verbe), cat (= non précisé) »

- 024, 029, 034, 039 : ces champs donnent les modalités d'usage du terme anglais. En première partie, ils précisent si le terme est en anglais « be » ou en américain « ae » (si la valeur est « be_vs_ae », la donnée n'est pas fournie); ensuite ils fournissent un commentaire sur la précision de la traduction dont la valeur par défaut est « comment » ; enfin, ils déterminent le domaine d'utilisation ou de non utilisation de la traduction qui vaut par défaut « restriction ».
- 025, 030, 035, 040 : ces champs ne sont pas encore utilisés.

5.1.3. Les champs de 041 à 060

acompte
masc
subst
comment + restriction
045
-
gen
cat
comment + restriction
050
-
gen
cat
comment + restriction
055
-
gen
cat
comment + restriction
060

Ces champs concernent les informations relatives à la traduction française.

- 041, 046, 051, 056 : ces champs contiennent au moins une traduction française du concept. La valeur « - » signifie « pas de traduction ».
- 042, 047, 052, 057 : ces champs précisent le genre du terme français : « (masc ; fem), fem, fem plur, masc, masc plur, gen (= non précisé) ».
- 043, 048, 053, 058 : on retrouve dans ces champs la catégorie syntaxique du mot français : « adj, expr, p-expr (= expression commençant par une préposition), subst, verb, cat (= non précisé) »
- 044, 049, 054, 059 : ces champs donnent les modalités d'usage du terme français. En première partie, ils fournissent un commentaire sur la précision de la traduction dont la valeur par défaut est « comment » ; ensuite, ils déterminent le domaine d'utilisation ou de non utilisation de la traduction qui vaut par défaut « restriction ».
- 045, 050, 055, 060 : ces champs ne sont pas encore utilisés.

5.1.4. Les champs de 061 à 080

aanbetaling
(de;het)
subst
064
afkorting
-
gen
cat
069
070
-
gen
cat
074
075
-
gen
cat
079
080

Ces champs concernent les informations relatives à la traduction néerlandaise.

- 061, 066, 071, 076 : ces champs contiennent au moins une traduction néerlandaise du concept. La valeur « - » signifie « pas de traduction ».
- 062, 067, 072, 077 : ces champs précisent le genre du terme néerlandais : « (de ; het), de, het, meervoud, gen (= non précisé) ».
- 063, 068, 073, 078 : on retrouve dans ces champs la catégorie syntaxique du mot néerlandais : « adj, expr, p-expr (= expression commençant par une préposition), subst, verb, cat (= non précisé) »
- 065 : ce champ procure une abréviation de la première traduction néerlandaise du concept. Si cette abréviation n'est pas fournie, la valeur de ce champ vaut « afkorting ».
- 064 ,069, 070, 074, 075, 079, 080 : ces champs ne sont pas encore utilisés.

On remarque ici que l'on duplique de l'information. En effet, la première traduction néerlandaise du terme correspond au champ 001.

5.1.5. Les champs de 081 à 100

Anzahlung
die
subst
comment + restriction
085
Abschlagszahlung

die
subst
comment + restriction
090
-
gen
cat
comment + restriction
095
-
gen
cat
comment + restriction
100

Ces champs concernent les informations relatives à la traduction allemande.

- 081, 086, 091, 096 : ces champs contiennent au moins une traduction allemande du concept. La valeur « - » signifie « pas de traduction ».
- 082, 087, 092, 097 : ces champs précisent le genre du terme allemand : « (der ; das), (der ; die ; das), das, der, die, plur, gen (= non précisé) ».
- 083, 088, 093, 098 : on retrouve dans ces champs la catégorie syntaxique du mot allemand : « adj, expr, p-expr (= expression commençant par une préposition), subst, verb, prefix, cat (= non précisé) »
- 084, 089, 094, 099 : ces champs donnent les modalités d'usage du terme allemand. En première partie, ils fournissent un commentaire sur la précision de la traduction dont la valeur par défaut vaut « comment » ; ensuite, ils déterminent le domaine d'utilisation ou de non utilisation de la traduction qui vaut par défaut « restriction ».
- 085, 090, 095, 100 : ces champs ne sont pas encore utilisés.

5.2. Le fichier ABB

Ce fichier comprend 6 775 entrées multilingues. Il ressemble fortement au fichier Spectrum. Chaque entrée est divisée en autant de champs. Ici, seule une traduction française est disponible pour chaque concept néerlandais. Ce fichier ne comprend donc pas de traduction anglaise et allemande.

Seuls les champs correspondant aux traductions néerlandaises (de 061 à 080) diffèrent par rapport à Spectrum.

aanbesteder
m
subst
znl_vs_nnl
afkorting
-
gen
cat
znl_vs_nnl
070
-
gen
cat
znl_vs_nnl
075
-
gen
cat
znl_vs_nnl
080

Ces champs concernent les informations relatives à la traduction néerlandaise.

- 061, 066, 071, 076 : ces champs contiennent une seule traduction néerlandaise du concept. La valeur « - » signifie « pas de traduction ».
- 062, 067, 072, 077 : ces champs précisent le genre du terme néerlandais : « m, v, o, (m ; v), (m ; o), (v ; m), (v ; o), (o ; m), meervoud, gen (= non précisé) ».
- 063, 068, 073, 078 : on retrouve dans ces champs la catégorie syntaxique du mot néerlandais : « (adj ; adv) (= si adjectif ou adverbe), subst, verb, cat (= non précisé) »
- 064, 069, 074, 079 : ces champs précisent si le terme est en néerlandais du sud des Pays-Bas « znl » ou du nord des Pays_Bas « nnl ». « znl_vs_nnl » indique que cette donnée n'est pas présente.
- 065 : ce champ procure une abréviation de la première traduction néerlandaise du concept. Pour l'instant, ce lexique n'en contient aucune et la valeur de ce champ vaut donc « afkorting ».
- 070, 075, 080 : ces champs ne sont pas encore utilisés.

On remarque que ces lexiques contiennent bien l'information souhaitée d'après la classification de la section 4.

- pas d'indication de prononciation
- pas d'étymologie
- plus d'information sur les modalités d'usage pour indiquer la discipline dans laquelle le mot est utilisé. Cette information se retrouve dans les champs « comment » et « restriction »
- pas de référence à la conjugaison ou à la déclinaison
- Le genre grammatical et la catégorie sont occasionnellement indiqués. Ici elles le sont par les champs consacrés au genre et à la catégorie.
- Moins de sens, mais il est spécifié par les définitions. Dans notre lexique, on trouve des définitions. Le champ « acceptation » donne les différents sens d'un même concept.
- Les synonymes et antonymes ne sont pas considérés comme indispensables. Ici, on retrouve les synonymes par les différentes traductions possibles, quatre au maximum.
- Les idiomes et contextes ne figurent que dans les encyclopédies.
- La traduction du mot d'une langue vers d'autres. Ce lexique propose la traduction du concept en quatre langues : anglais, français, néerlandais et allemand et pour chaque langue, on peut avoir quatre traductions différentes.

A partir de ces lexiques, on a conçu pour chaque fichier une base de données, de manière telle qu'elle réponde au besoin du dictionnaire multilingue d'Apollo, mais aussi à n'importe quel autre dictionnaire multilingue. Cette base de données, décrite au chapitre 4, doit donc contenir toute l'information qu'un tel dictionnaire présente généralement et cette information doit être structurée de manière à ce qu'on puisse y accéder rapidement.
Le système possède donc deux bases de données de même format. On a évité de regrouper les deux fichiers dans une seule base de données pour la raison déjà invoquée plus haut, à savoir la difficulté de correspondance entre les mêmes termes se trouvant dans les deux fichiers différents.
Le développement de la conception et l'implémentation de la base de données sont précisés dans le chapitre 4.

Chapitre 2. Cadre méthodologique

1. INTRODUCTION

Le projet Apollo a pour objectif de fournir un banc de travail pour la création et la maintenance de documents multilingues dans les secteurs bancaire et financier. Il permet donc par exemple, d'afficher un document français et son correspondant anglais sur un même écran. Il peut également gérer les différents versions des documents multilingues. Ce projet européen est détaillé dans le chapitre 3.
Il a donc pour objet la traduction automatique et requiert, comme tout traducteur, un corpus dans lequel il puise les réponses à ses besoins lexicographiques.
Ce corpus a été créé par le partenaire FUNDP-ELV du projet Apollo, qui avait en charge tous les travaux lexicographiques.
Ce chapitre se consacre à l'étude de ce corpus. Nous allons examiner les ressources qui ont permis de le créer, ainsi que les applications qui l'utilisent.

2. UN PEU DE THEORIE

Avant d'analyser le corpus spécifique d'Apollo, il me paraît intéressant de reprendre quelques considérations théoriques proposées par Gunnel Engwall [ATKINS] concernant l'élaboration générale d'un corpus. Il fait d'abord quelques considérations générales comme la sélection des textes et leur représentation électronique, puis il étudie plus précisément les caractéristiques des textes pouvant servir à la conception d'un corpus.

D'après cet auteur, la première chose essentielle à déterminer est le domaine pour lequel on veut créer un corpus. Il s'agit donc de dresser les frontières qui délimiteront les branches que l'on désire analyser avec le corpus que l'on crée. Ce n'est qu'à ce moment que l'on pourra vérifier si un texte est susceptible d'appartenir au corpus. L'objectif de l'étude étant bien défini, il permet de planifier les recherches.
Ensuite, il convient de tenir compte des ressources disponibles. Le projet va dépendre de ressources telles que le temps, les fonds, la main d'oeuvre,... Elles vont parfois entraîner des restrictions du domaine que l'on souhaitait analyser.

Gunnel Engwall prétend que la taille du corpus dépend de son utilité : il doit être plus large s'il est utiliser comme base à un dictionnaire général et se rétrécit s'il sert à l'étude d'un domaine spécialisé. Cependant, il n'existe pas de critère pour déterminer cette taille; elle doit se décider en fonction des désirs et des ressources.

2.1. La sélection des textes

Les textes sélectionnés pour la constitution d'un corpus donnent non seulement les mots utilisés dans un domaine, mais également leurs modes d'utilisation.
Selon l'auteur, la sélection des textes utilisés pour la conception d'un corpus ne se fait pas au hasard. Au contraire, elle se fait par étapes comme le montre le schéma suivant:

catégorie → genre → période → texte → exemple

Il faut d'abord choisir la catégorie des textes que l'on veut observer : se concentre-t-on sur des textes littéraires, scolaires, sur des journaux ou sur des conversations. On peut choisir plus d'une catégorie, mais il faut être conscient qu'on ne pourra pas analyser tous les types de textes étant donné l'énormité du travail que cela entraîne. On peut déjà remarquer ici que les textes peuvent être de source écrite ou orale.
Le choix suivant à faire concerne le genre des textes : prose, drame, textes scientifiques, dialogue,...
Ensuite, il faut situer l'avancement des recherches dans le temps et ainsi établir un planning.
Finalement, on peut décider quels textes ou paragraphes de textes vont aider à l'élaboration du corpus.
Le choix de ces échantillons de textes affecte le résultat des recherches. En effet, si le même nombre de mots est extrait dans chaque texte, il sera plus facile de comparer ceux-ci quantitativement que si l'on y pêche un nombre quelconque de mots.
De plus, le nombre de mots distincts intéressants à retenir dans un texte décroît si la longueur des textes augmente. En effet, plus le texte est long, moins on retrouve de mots distincts.
Le choix de la longueur des textes est donc un point important à prendre en compte lorsqu'on réalise un corpus.

2.2. La représentation électronique des textes sélectionnés

Une fois les textes sélectionnés, qu'ils soient écrits ou oraux, ils doivent être transformés sous forme électronique. Celle-ci doit comprendre toutes les données que nous fournissent les mots

du texte : les noms propres, le genre et le nombre des mots doivent être respectés. Si ces mots sont des verbes conjugués, il faudra également en tenir compte.
Il faut parfois décoder le contenu des textes pour pouvoir l'utiliser. C'est le cas lors de l'utilisation de textes techniques fournis par des entreprises utilisant certains identifiants ou autres codes.
Il est évidemment plus facile de rendre électronique une ressource écrite qu'une ressource orale. En effet, celle-ci exige un enregistrement puis une retranscription si elle se veut précise.

2.3. Le problème du copyright

Le problème des droits de propriété concerne toute personne qui crée un corpus. Il faut vérifier que l'on possède les droits de pouvoir collecter des textes avant de commencer toute procédure.

2.4. Les caractéristiques du texte

Il faut tenir compte d'une série de caractéristiques des textes lors de leur sélection. Ces caractéristiques vont permettre de classifier les différents textes, mais aussi les divers corpus conçus à partir de ces textes.

- la forme : Un texte peut être écrit ou oral. S'il est écrit, il peut l'être manuellement ou électroniquement. Lorsque la source est orale, il peut s'agir d'une conversation personne à personne, d'un discours, d'une conversation téléphonique ou télévisée,...
- le contenu : S'agit-il de textes littéraires qui traitent de fiction, ou scolaires, traitant alors de faits? Cette distinction entre fiction et fait n'est pas toujours évidente à percevoir.
- la continuité : Cette caractéristique indique si le texte apparaît régulièrement dans un magazine ou dans un quotidien. Il s'agit alors de textes « continus ». Un texte est dit discontinu lorsque c'est une lettre personnelle, un acte d'une pièce de théâtre,... Ces sont alors des textes non périodiques.
- la préparation : Un texte est préparé s'il a été publié après avis et correction. Dans le cas contraire, on parle de textes spontanés. Un texte rédigé est donc en général préparé, tandis qu'une conversation est spontanée.
- la disponibilité : Ici, on caractérise l'accessibilité qu'on a, par rapport à un texte. Le texte écrit est en général plus rapidement disponible que la source orale.
- la délimitation : Le contenu du texte est soit bien déterminé et délimité, soit sans frontière distincte. Le texte écrit est mieux délimité que l'expression orale.

Ces caractéristiques peuvent toutes être nuancées. Il est par exemple parfois difficile de distinguer ce qu'est un fait et ce qu'est une fiction ou de percevoir dans quelles mesures un texte est périodique ou non,...

2.5. Les catégories des textes

Les corpus peuvent être de différents types, selon qu'ils se basent sur des catégories de textes différentes. Nous allons détailler les caractéristiques de ces corpus et des textes qui ont aidé leur élaboration.

2.5.1. Les travaux littéraires et scolaires

Les travaux littéraires et scolaires sont généralement d'un genre prestigieux, sophistiqués et abordent un grand nombre de sujets. C'est pourquoi ils sont souvent sélectionnés pour faire partie de corpus linguistiques.
Les textes scolaires quant à eux font souvent partie d'un corpus contenant plusieurs genres. On se sert de ces textes pour trouver des terminologies exactes dans différents domaines.

De quelle **forme** sont ces textes ? Ils sont écrits et publiés dans des unités distinctes comme des volumes d'ouvrage par exemple. Ces volumes sont structurés en sections, chapitres ou paragraphes. De cette façon, on pourrait imaginer que le livre est un domaine à étudier et que le chapitre est un échantillon choisit pour intégrer dans le corpus. Les frontières quant à elles pourraient être fixées aux oeuvres écrites par l'auteur de ce livre.
Ces textes incluent différents genres, tels que des nouvelles, des poèmes, des drames, mais aussi des publications physiques, technologiques, économiques, légales ou médicales.
Le **contenu** des textes littéraires est souvent de la fiction, tandis que les textes scolaires proposent des faits. Ceci implique que la littérature ne se démode pas, mais qu'au contraire, les travaux scolaires doivent évoluer et être réédités sous de nouvelles versions pour rester au goût du jour. Les textes scolaires présentent souvent leur contenu à l'aide d'illustrations, de tables,... Ici se pose le problème d'intégrer ces informations dans le corpus. Des décisions doivent être prises à ce sujet et elles varieront en fonction de l'objectif de corpus.
Ces types d'oeuvres sont considérés comme étant **discontinus**. En effet ils ne sont pas systématiquement publiés à des intervalles réguliers. Il devient donc difficile de les comparer à travers le temps.
La **préparation** d'un tel texte est souvent imposante. En littérature, l'auteur fait beaucoup de révisions de son texte avant de le publier. Au niveau scolaire, ces révisions sont également

présentes mais il y en a moins. En effet, les nouvelles versions apparaissent rapidement et on exigent donc pas la perfection que requièrent les oeuvres littéraires.
La **disponibilité** de ces textes est assez élevée. Chaque oeuvre littéraire a un numéro d'identification et est enregistrée dans les librairies nationales. On peut donc facilement y accéder sur demande.
Ces textes posent des problèmes de **délimitation**. En effet, on a déjà vu qu'ils concernent différents domaines et qu'ils peuvent appartenir à différents genres. Cependant, les idées qui y sont énoncées sont en général bien délimitées.

Ce genre de ressources a servi par exemple à établir la compilation « Trésor de la langue française ». Ce corpus permet d'étudier des auteurs comme Emile Zola ou Victor Hugo. Il contient 70 millions de mots.

2.5.2. Les journaux

Lorsqu'on veut élaborer un corpus linguistique général en insistant sur le vocabulaire et la grammaire, les journaux donnent une meilleure base que les oeuvres littéraires. Ils facilitent l'analyse de la langue naturelle.
Ils ne sont cependant pas homogènes et abordent beaucoup de sujets.
Par journaux, on entend les quotidiens, les revues hebdomadaires, mensuelles, ainsi que des publications un peu moins fréquentes.
Les articles de journaux sont quotidiens ou hebdomadaires, nationaux ou régionaux, écrits sous une influence politique ou indépendants,... Tous ces articles doivent être classifiés pour la sélection et ce n'est pas toujours tâche aisée.
Ces publications sont de **forme** écrite.
Elles **contiennent** soit de la fiction soit des faits. Les sujets abordés sont souvent répétés. Ces textes comprennent des éléments qu'on ne retrouve que très peu dans les autres oeuvres comme des publicités, des photos, des images,... Il est donc important de décider, lors de l'élaboration du corpus, si on intègre ou non ces éléments et si oui, comment ils vont être gérés.
Les journaux sont **continus**. Ils sont planifiés et publient des textes selon une certaine fréquence. De bonnes comparaisons à travers le temps sont possibles.
Les textes de la presse sont également des textes **préparés**. Mais la préparation est beaucoup plus courte. Les articles sont rédigés rapidement. Ensuite, ils sont soumis à une autorité qui décide les parties de l'articles qui vont être publiées. Parfois, en raison de l'actualité, tout l'article peut être rejeté. Le contenu exact de l'article n'est parfois décidé que tardivement dans le processus de rédaction.
Un journal doit être **disponible** par définition. De plus les rédacteurs des journaux sont en général prêts à fournir leurs articles sous forme électronique. Cependant, lors de la sélection

des textes à insérer dans le corpus, il faut être conscient des conditions dans lesquelles ils ont été écrits. Ils sont parfois écrits sous une influence politique ou autre.
Etant une source écrites, les textes de journaux peuvent être considérés comme étant **délimités**. Leur portée est en général assez précise.
On trouve plusieurs corpus de journaux ces dernières années : des corpus de journaux suédois, allemands, espagnols ou français. Tous ces travaux analysent le langage contemporain et étudient le vocabulaire généralement utilisé.

2.5.3. La correspondance et les sources orales

2.5.3.1. La correspondance

La **forme** de la correspondance est par définition l'écrit. Cet écrit implique un expéditeur et un ou plusieurs destinataires.
En général, le **contenu** d'une lettre est assez caractéristique : un tel texte est daté, commence par une forme de salutation et se termine par une formule de politesse. Il peut contenir des faits ou de la fiction.
Les lettres sont **discontinues**. Un même expéditeur peut en écrire plusieurs sans que ce phénomène ne soit considéré comme continu.
Du point de vue de la **préparation**, la correspondance peut être revue plusieurs fois avant d'être envoyée, mais elle peut aussi être spontanée et ainsi être envoyée directement après sa rédaction, sans relecture.
La correspondance n'est pas facilement **disponible**, non seulement parce qu'on ne conserve pas toujours les lettres, mais en plus, parce qu'elles sont dédiées à un public restreint qui n'a pas toujours l'intention d'en communiquer le contenu.

On a par exemple réalisé des travaux sur les lettres écrites par August Strindberg. D'autres ont étudié la correspondance française dans les mouvements politiques de mai 68.

2.5.3.2. Le monologue

Le monologue est, comme son nom l'indique, de **forme** orale.
Son **contenu** est assez vaste et peut comprendre de la fiction ou des faits. Le monologue peut en effet être imaginatif, quand je me dis quelque chose ou bien informatif, lorsque je fais une présentation devant un groupe.
Il est souvent **discontinu**. Cependant, certaines présentations télévisées se font toutes les semaines et peuvent donc être considérées comme continues.
En général, dans un monologue, on retrouve des parties **préparées** et d'autres spontanées, et c'est souvent ces parties qui font le charme de la présentation.

La **disponibilité** et la **délimitation** de telles sources sont faibles. En effet, la seule façon de rendre un monologue accessible, c'est de l'enregistrer. Or cette méthode n'est pas souvent utilisée. Dès que la présentation est terminée, on en a oublié une partie du contenu et il devient alors difficile de le délimiter.
Le problème majeur que pose le monologue lors de la création d'un corpus est son mode de représentation électronique. Comment peut-on représenter un discours de façon électronique ? Il faut d'abord l'enregistrer puis le retranscrire. Il est donc rarement possible de faire des analyses phonétiques sur base de ces ressources.

Une analyse a été faite par Cotteret et Moreau. Ils ont étudié les discours radio du Général de Gaule de 1958 à 1965.

2.5.3.3. Le dialogue

Ici aussi il s'agit bien évidemment d'une **forme** orale. Ces conversations peuvent se faire à distance ou face à face.
Le **contenu** est à nouveau très diversifié et hétérogène, avec des faits ou de la fiction.
Ces dialogues ne sont pas **continus**, étant donné que les participants à ce dialogue changent en permanence.
Ils sont presque toujours **spontanés**.
Tout comme pour le monologue, la **disponibilité** et la **délimitation** sont faibles.

Le même problème de représentation de la source orale persiste. Cependant, ces sources sont intéressantes pour les linguistes qui désirent créer un corpus multilingue par exemple. N'est-il pas important d'étudier la forme d'expression la plus spontanée ?

Le « London-Lund Corpus of Spoken English » contenant des textes oraux de l'étude « Survey of English Usage » (Svartvik et Quirk 1980; Svartvik 1990), reprend un demi million de mots trouvés dans des conversations spontanées entre des interlocuteurs anglais.

2.5.4. En résumé

catégorie	forme	contenu	continuité	préparation	disponibilité	délimitation
littéraire	écrit	fiction	faible	élevé	élevé	élevé
scolaire	écrit	faits	faible	élevé	élevé	élevé
journal	écrit	mixte	élevé	élevé	élevé	élevé
correspondance	écrit	mixte	faible	mixte	faible	faible
monologue	oral	mixte	faible	mixte	faible	faible
dialogue	oral	mixte	faible	faible	faible	faible

Il est évidemment plus aisé de travailler avec un texte écrit qu'avec un texte oral. Un discours préparé est plus facile à utiliser qu'une discussion spontanée. Cependant, on s'est de plus en plus attardé aux sources orales car elles sont pertinentes pour l'étude de la linguistique.
Le contenu est très souvent fictif et factuel. Dans beaucoup de textes de n'importe quelles catégories, on trouve des parties imaginatives, puis d'autres qui relatent des faits.
Un haut degré de continuité permet de choisir des échantillons selon un planning. Sachant quels articles vont se présenter dans le temps, on peut choisir « à l'avance » les textes à intégrer dans le corpus.
En sélectionnant des textes écrits, on a de fortes chances pour que ces textes soient préparés. Ils ont bien souvent été relus. Par contre, dans le dialogue de famille ou entre amis, le degré de préparation est quasi nul.
Un texte écrit est plus facilement accessible et utilisable qu'une conversation recopiée. Ceci était déjà vrai avant les ordinateurs et reste vrai depuis l'avènement de l'informatique, les corpus se faisant souvent à l'aide de cet outil. Le scanner aide la promotion des sources orales, mais les systèmes de reconnaissance de la parole apporteront un changement radical dans ce domaine. Le problème de la représentation de la source électroniquement restera cependant le même, et fera l'objet de nouvelles analyses.
Si l'on travaille avec des textes dont les frontières sont bien définies, on rencontre moins de problèmes que si le texte n'est pas délimité et traite trop d'aspects différents. Ainsi, le dialogue spontané est une fois de plus difficile à maîtriser.

3. LE CORPUS APOLLO

Sur base de cette théorie, essayons maintenant d'analyser le corpus Apollo. Il se base sur des textes existant et sur des ressources lexicographiques.

3.1. Les ressources textuelles

Le projet Apollo limite son analyse au domaine bancaire et financier. Son corpus a été créé sur base de trois ressources textuelles. Celles fournies par ABB et par IFBL consistent en une série de cours boursiers dans leur format original; elles sont donc standards. C'est à dire qu'elles ne sont pas structurées et ne contiennent aucune marque. Par contre la ressource proposée par INIST est structurée. Il s'agit de références bibliographiques de livres, d'articles ou de textes. Chaque fiche reprend le titre, l'auteur et un résumé du texte ainsi que des informations

permettant de les classifier telles que le domaine, le type de document ou le langage. La ressource INIST se compose de données extraites d'une base de données stockées dans différents champs.

Voici un exemples de données fournies par INIST. Dans cet exemple, Les informations item_id, langue_resume et numero ont été ajoutées par le partenaire FUNDP-ELV.

- ITEM_ID : INIST_FR_016_616.91.0064
- Domaine : gestion
- Titre : Changement technique, innovation produit et performance socio-économique en agences bancaires
- Auteur : LALLE B
- Adresse : -
- Organisme : -
- Type_doc : Thèse
- Source : FRA; 493 p., ill., tabl., ann.; bibliogr.; ABS. fre; - Thèse de Doctorat Sciences de gestion; SAVALL (H.), dir.; FRANCE. Institut de Socio-Economie des entreprises et des organisations. (ISEOR). Ecully; DA. 1989/12/18
- Langue : Français
- Date_pub : 1989
- Résumé : Après avoir évoqué le changement technique et l 'évolution de l'environnement de l'agence bancaire, l'auteur aborde le problème du passage à l'innovation produit avec le concept de 'produit caché'. Place de ces produits dans la stratégie bancaire, et proposition de valorisation de ces produits.
- LANGUE_RESUME : FR
- Code_class : 616-2
- Mots_clés : agence bancaire; télématique; performance socio-économique; paiement électronique; France - bank branch; telematics; socio-economic efficiency; electronic payment; France
- Localisation : INIST; ISEOR : 40 T 19 Numéro : 616.19.0064
- NUMERO : 616.91.0064

Ces trois ressources textuelles pourraient être considérées comme des textes scolaires. En effet, elles sont écrites et relatent des faits. Elles ne sont pas continues. Ces ressources demandent une grande préparation et leur structuration en est la preuve. Elles sont disponibles à plusieurs applications. Grâce à leur classification, elles sont également bien délimitées pour le projet Apollo aux fiches concernant les domaines bancaire et financier.

Qu'elles soient structurées ou non, toutes les ressources textuelles ont été réorganisées pour qu'elles deviennent homogènes et pour qu'elles puissent ainsi faire partie d'un même corpus .

Pour délimiter le corpus, un ensemble de paragraphes ont été sélectionnés dans les modules consacrés aux bourses. Ils serviront à modéliser le domaine dans lequel on travaille. Ces paragraphes ont été choisi selon certains critères : ils ont une longueur minimale et ne contiennent pas de caractères spéciaux.

Corpus ABB	Paragraphes reçus	Mots	Mots différents	Paragraphes exemples
Module A	887	18654	3237	10
Module B	1169	26467	4199	10
Module C	421	7530	1716	10

Corpus IFBL	Paragraphes reçus	Mots	Mots différents	Paragraphes exemples
Module 1	508	12377	2896	10
Module 2	600	11999	2748	10
Module 3	983	9983	1827	10
Module 4	579	12040	2333	10
Module 5	1340	22095	3843	10
Module 6	1237	20577	3577	10

Corpus INIST	Paragraphes reçus	Mots	Mots différents	Paragraphes exemples
Français	166	12194	1328	10
Anglais	56	3895	2975	10

Ressources textuelles utilisées dans Apollo

3.2. Les ressources lexicographiques

Comme je l'ai déjà mentionné dans le chapitre précédent, Les partenaires Spectrum et ABB ont fournit des lexiques.

En effet, Spectrum possède un lexique contenant 1932 concepts bancaires et financiers : le « Prisma Vakwoordenboek bankzaken ». Après exploration de toutes ces entrées, seules 1365 d'entre elles ont été retenues, soit 70.6%.

ABB, quant à lui, nous a fournit le dictionnaire de la terminologie bancaire, soient 7105 concepts, parmi lesquelles 6775, soit 95.3% ont été gardées.

Ces ressources pourraient également être qualifiées de ressources scolaires : elles sont écrites, donnent des faits. Elles exigent une préparation et sont disponibles. Leur secteur est limité au bancaire et au financier.

Pour compléter encore le corpus, certaines données manquantes dans ces ressources, telles que les monnaies, ont été recherchées dans le réseau « world wide web ». Ainsi, le site « currencies of the world » nous a fourni les monnaies utilisées dans 126 pays du monde [http://pacific.commerce.ubc.ca/trade/currencies.html]. Ces éléments ont été organisés en un lexique de monnaies.

Certaines données ont été rajoutées manuellement. Par exemple, si un mot français a comme catégorie syntaxique « substantif », la catégorie syntaxique du mot anglais ou néerlandais sera également « substantif ».
Certaines corrections ont dû être faites pour adapter ou ajouter de l'information.

Le projet Apollo a suivi à peu près la démarche proposée par Gunnel Engwall. En effet, dans l'objectif même du projet, on a tout de suite précisé le domaine avec lequel on voulait travailler. Il est évident que pour traiter le secteur bancaire et financier, ce ne sont pas des textes littéraires qui vont être utiles mais plutôt des textes scientifiques plus spécifiques. Tous les travaux lexicographiques ont été planifiés dès le début du projet. La recherche des textes s'est faite assez rapidement à l'aide de certains partenaires prêts à donner leurs ressources. Des paragraphes ont été sélectionnés selon certains critères expliqués précédemment.

3.3. Remarques sur la qualification

Dans la section précédente, on a classifié les ressources du corpus Apollo comme étant scolaires. En effet, si on suit ce que nous propose Gunnel Engwall [ATKINS], par élimination, on serait tenté de les ranger dans cette catégorie.
Pourtant, ces ressources sont des textes techniques qui nous ont été fournis par des spécialistes. Ils ont donc un statut spécifique et pourraient faire partie d'un sous-langage comme il est définit par [DEVILLE] : « a sublanguage is a set of utterances referring to a limited and well-defined semantic domain and used for a specific function. Such utterances are generated by a specific grammar and a specific vocabulary. »

En effet, nos textes appartiennent à un domaine bien spécifique qui est le domaine financier et bancaire. Certaines ressources sont utilisées uniquement comme références bibliographiques et ont donc un but bien précis. On peut également considérer qu'ils utilisent une grammaire bien spécifique vu qu'ils traitent par exemple de cours de bourse ou de fiches signalétiques contenant des champs et des concepts qu'on n'utilise pas en parlant.
C'est ainsi que les textes constituant le corpus Apollo pourraient faire partie d'un sous-langage spécifique au secteur bancaire et financier.

3.4. Utilisateurs de ces ressources

Ces ressources ont été modélisées par FUNDP-ELV et vont être utilisées par le traducteur automatique et par le dictionnaire multilingue.

3.4.1. CAT2 : le système de traduction automatique

Dans le projet Apollo, on se contente d'utiliser le traducteur automatique pour la paire de langue anglais-français. Les ressources lexicales sont à la base du système de traduction automatique.
Plusieurs autres outils supplémentaires ont été développés à partir de ces lexiques pour CAT2. FUNDP-ELV a par exemple établi des listes de termes anglais accompagnés du concept leur correspondant en néerlandais et la même chose a été réalisé pour les termes français. Ces listes permettent à CAT2 de générer automatiquement des entrées lexicales anglaises et françaises.
On a donc établi deux listes binaires. La liste anglaise contient 1 571 termes et la française en contient 1 497.

Voici un exemple de la liste des termes anglais :

ENGLISCHES_WORT down payment NIEDERLAENDISCHES_KONZEPT aanbetaling
ENGLISCHES_WORT invest NIEDERLAENDISCHES_KONZEPT beleggen
ENGLISCHES_WORT dull NIEDERLAENDISCHES_KONZEPT flauw

Voici un exemple de la liste des termes français :

FRANZOESISCHES_WORT acompte NIEDERLAENDISCHES_KONZEPT aanbetaling
FRANZOESISCHES_WORT investir NIEDERLAENDISCHES_KONZEPT beleggen
FRANZOESISCHES_WORT placer NIEDERLAENDISCHES_KONZEPT beleggen
FRANZOESISCHES_WORT terne NIEDERLAENDISCHES_KONZEPT flauw
FRANZOESISCHES_WORT inactif NIEDERLAENDISCHES_KONZEPT flauw

3.4.2. Le dictionnaire multilingue

Dès la réception du corpus, les ressources lexicales ont été organisées dans une base de données relationnelles explicitée au chapitre 4. Toutes les données présentes dans les lexiques y ont été insérées. Le dictionnaire accède à ces informations et les affichent à l'écran au moyen d'une interface.

Ces utilisateurs, CAT2 et le dictionnaire multilingue, seront explicités dans le chapitre suivant.

Chapitre 3. Cadre opératoire : Le projet APOLLO

1. INTRODUCTION

Ce chapitre a pour objet l'étude du cadre dans lequel s'insère le dictionnaire multilingue : le projet Apollo. Nous allons d'abord étudier les principales composantes du projet et nous verrons ensuite comment le dictionnaire multilingue s'introduit dans ce projet.

2. QU'EST-CE QUE LE PROJET APOLLO?

2.1. Introduction

On se référera aux rapports d'activités produits par les différents partenaires du projet pour une description détaillée d'Apollo. Ces rapports peuvent être consultés à partir du World Wide Web à l'adresse suivante : « http://www.crpcu.lu/APOLLO ». Ce sont ces rapports qui ont inspirés la description suivante du cadre opératoire.
Apollo signifie « An oPen wOrkbench for muLtiLingual dOcument creation and maintenance».
Le but principal de ce projet est de fournir un banc de travail pour la création et la maintenance de documents multilingues dans le secteur bancaire et financier. Il veut donc d'offrir la possibilité de traduire à un même moment un texte en plusieurs langues et d'afficher ces différentes traductions simultanément.
Apollo est un projet européen qui a une durée de 15 mois.

Il a paru utile de travailler en différentes étapes. En effet, on a choisi de développer dans un premier temps une maquette avant de réaliser un banc de travail complet.
La maquette ne traite que la paire de langues français - anglais. Elle sert à montrer aux utilisateurs potentiels les possibilités dans ce domaine. Elle n'est donc pas commercialisable.

Le projet se résume à trois principaux objectifs :

- spécifier et identifier clairement les besoins de l'utilisateur dans le domaine multilingue.
- développer une maquette pour montrer les possibilités de gestion des documents multilingues.
- étudier le planning de travail pour établir un banc de travail complet, c'est à dire définir la suite du projet Apollo, soit Apollo2.

Ce projet a rassemblé dix partenaires dans cinq pays européens différents. En voici la liste ainsi qu'une brève description de leurs rôles respectifs.

En Belgique :

- Association Belge des Banques (ABB) : Ils ont été les utilisateurs du projet et comme on l'a déjà dit plus haut, ils ont fourni des ressources linguistiques pour réaliser le projet.
- Facultés Universitaires Notre-Dame de la Paix - Ecole des Langues Vivantes (FUNDP-ELV) : Ils ont en charge les développements linguistiques.
- Université de Liège - Service de Technologie de l'Education (STE) : Ils ont fait le lien entre le projet et les utilisateurs potentiels, notamment par une étude des besoins.

En France :

- Institut de l'Information Scientifique et Technique (INIST) : leur rôle est le même que celui d'ABB; en effet, ils sont utilisateurs du projet et fournisseurs de ressources linguistiques.

En Allemagne :

- Institut der Gesellschaft zur Förderung der Angewandten Informationsforschung (IAI) : Ils se sont occupés des développements linguistiques et ont fournit un traducteur automatique.

Au Luxembourg :

- Centre de Recherche Public - centre Universitaire (CRP-CU) : Ils sont fournisseurs de technologies.
- Institut de Formation Bancaire, Luxembourg (IFBL) : Ils jouent le rôle d'utilisateurs. Ils ont également fourni des ressources linguistiques déjà mentionnées plus haut.
- Office Future International Services (OFFIS) : Ce partenaire est aussi un fournisseur de ressources technologiques. Il s'est occupé de la gestion des versions des documents multilingues.
- SILIS s. à r. l. (SILIS) : Ils ont fourni un traitement de textes pour travailler les documents multilingues.

En Hollande :

- Uitgeverij Het Spectrum (SPECTRUM) : Ils fournissent également des ressources linguistiques dont j'ai déjà parlé.

Apollo se base sur un maximum de technologies existantes:

- La maquette est intégrée dans le traitement de texte Interscript (cfr. section 2.2.4.1.). Ce produit a été développé par Silis et est commercialisé dans différents pays européens. Ce traitement de texte a été modifié pour s'adapter aux besoins des documents multilingues.
- La maquette comprend des fonctions de traduction automatique. On se repose ici sur CAT2, un système de traduction automatique à base de transfert. Ce système a été développé par l'IAI. CAT2 (cfr. section 2.2.4.2.) offre une traduction robuste quand l'analyse fine a échoué. Il fournit donc, la plupart du temps, une traduction. Le système est implémenté en Prolog.

2.2. Les différentes tâches du projet

Le projet Apollo, pour réaliser ses objectifs, s'est assigné plusieurs tâches. En voici le détail.

2.2.1. L'étude de marché

Cette étude a examiné les besoins de différentes catégories d'utilisateurs dans les banques et le monde industriel et a recherché ce qui existe sur le marché pour répondre à ces besoins.

Les besoins ont été clairement définis et on a ainsi pu donner une ligne de conduite pour Apollo2.
Le STE a interrogé les banques ABB et IFBL et le secteur industriel via INIST et ses clients. L'étude s'est concentrée sur la France, la Belgique et le Luxembourg.
Les conclusions de l'étude des besoins montrent que l'utilisateur semble insatisfait de ce qui existe sur le marché. Il ne connaît que très peu d'outils qui existent dans ce domaine.
Le secteur industriel paraît plus intéressé par ce que propose Apollo. Il utiliserait un tel outil pour comprendre en quelques mots le contenu de documents techniques. Une traduction approximative suffit ici tandis que les banques exigent la plupart de temps une qualité de traduction parfaite. La traduction produite par Apollo ne leur semblerait certainement pas assez précise.

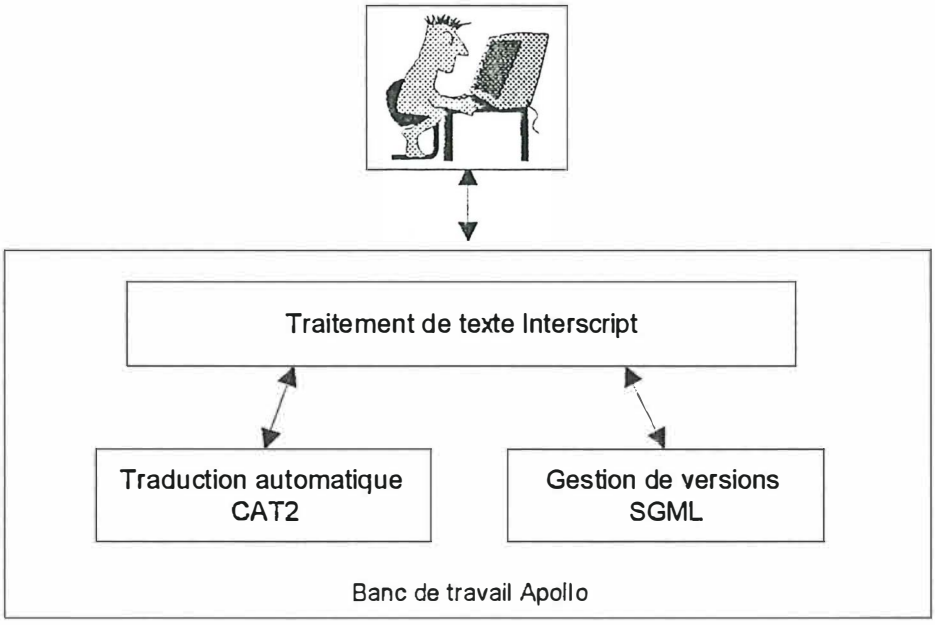
Quant à ce qui existe sur le marché, on observe différents produits tels que le traitement de texte, l'échange de fichiers, la gestion de version, le dictionnaire manuel et électronique, des

système de gestion de terminologie électronique, des mémoires de traduction, la traduction automatique, des bancs de travail intégrés,...
Il existe beaucoup de systèmes de traduction sur le marché actuellement, mais ceux-ci ne sont pas intégrés dans un traitement de texte. Le projet Apollo l'intègre dans Interscript. Ceci signifie que toute la structure du document doit être repensée. On veut par exemple pouvoir afficher un texte et sa traduction sur le même écran.
Il n'existe pas sur le marché de produit gérant les versions d'un document multilingue. Cet outils est proposé par Apollo.

2.2.2. La spécification du prototype

Le consortium a spécifié les fonctionnalités de la maquette mais aussi celles du projet Apollo2 grâce à l'étude des besoins des utilisateurs.
La maquette comprend plusieurs modules: un traitement de textes multilingue, un gestionnaire de versions SGML, un module de traduction automatique et un dictionnaire multilingue. Les trois premiers modules sont développés plus loin dans ce chapitre. Le dernier est décrit dans le chapitre 4.

Voici l'architecture du projet qui représente l'intégration des trois premiers modules.



Architecture du projet Apollo

L'utilisateur interagit uniquement avec le traitement de texte Interscript. Celui-ci fait appel aux autres modules pour offrir plus de services.

2.2.3. Les développements lexicographiques

Ces développements ont en grande partie déjà été expliqués dans les deux premiers chapitres. Nous avons déjà parlé du corpus qui a été créé pour ce projet ainsi que des modifications qui ont été apportées aux données. Résumons simplement ici ces développements.

Cette tâche consiste à collectionner les corpus et à modéliser le langage.

2.2.3.1. La collection d'un corpus

Les ressources textuelles provenant de ABB et IFBL ne possèdent pas de format spécifique. Elles consistent en une série de modules de bourses dans leur format original.
Par contre, celles provenant de INIST sont formalisées en différents champs. Il s'agit ici de fiches comprenant des références bibliographiques.
Toutes ces ressources ont été réorganisées.
Spectrum est spécialisé dans l'élaboration et la maintenance de ressources lexicographiques. Il a ainsi pu fournir un point de départ pour l'élaboration de lexiques multilingues.
Un autre dictionnaire ABB a été fournit sous format électronique.
Ces ressources lexicographiques ont également dû être réorganisées sous forme d'une base de données multilingue unique.
Un lexique de monnaies a également été développé sur base d'un site Web.

2.2.3.2. La modélisation du sous-langage

La base de données multilingue Apollo comprend 3 parties: le lexique Spectrum, le lexique ABB et le dictionnaire des monnaies.
Le néerlandais a été choisi comme interlangue, c'est à dire comme identificateur d'un concept.

Cette tâche s'est opérée en quatre phases :
1. observer : Quelles informations nous a-t-on fournies ?
2. vérifier et corriger : Ceci se fait par rapport à la structure d'information définie.
3. restructurer : Ici, il s'agit d'insérer les données fournies et corrigées dans la base de données
4. augmentation : On peut ensuite ajouter de l'information dans la base de données.

De nombreux autres travaux linguistiques ont été réalisés pour CAT2. Ainsi par exemple, chaque concept a été modélisé sous le format utilisé par CAT2. Pour ce faire, il a fallu ajouter de l'information à celle qu'on possédait manuellement. Les manipulations ont exigé beaucoup de temps mais ont facilité la tâche du traducteur.

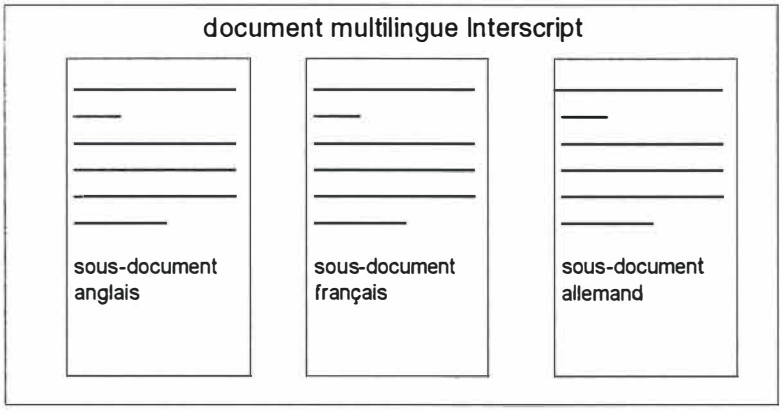
2.2.4. L'implémentation du prototype

2.2.4.1. Le traitement de texte Interscript

Interscript est le traitement de textes utilisé par Apollo. En effet, c'est à partir d'Interscript que l'utilisateur aura accès aux services de traduction automatique proposés par Apollo.
Ce traitement de textes a été adapté pour traiter les objets multilingues. Voici les fonctions qu'il propose dans ce domaine.

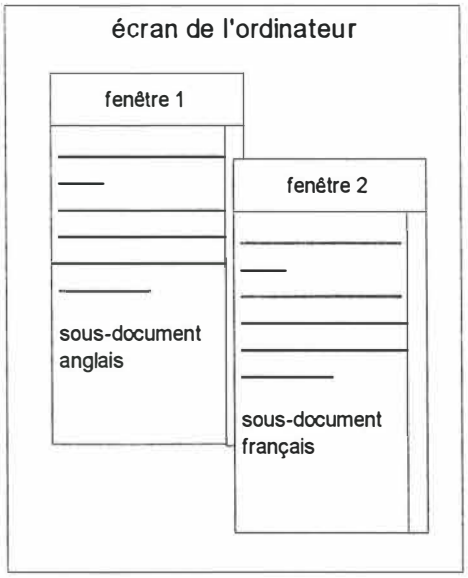
- Création et affichage d'un documents multilingues:

L'architecture d'un document Interscript a été modifiée. Un document multilingue est composé d'au moins un sous-document. Chaque sous-document contient le texte du document principal en une langue spécifique.
Admettons que l'on possède le mode d'emploi d'une machine X en français et sa traduction en anglais. Ce document multilingue « documentation de la machine X » possède donc un sous-document français, contenant le texte lui-même, ainsi qu'un sous-document anglais, contenant la traduction anglaise de ce texte.
Si l'on possède d'autres traductions de ce texte, elles seront reprises dans d'autres sous-documents.



Document multilingue Interscript

L'utilisateur peut créer un document multilingue, ajouter un sous-document à un document multilingue existant, supprimer un tel sous-document, éditer un sous-document.
Le traitement de textes peut afficher différentes fenêtres contenant différents sous-documents d'un document multilingue à l'écran.
On verra donc affichées à l'écran une fenêtre contenant le texte français et une autre contenant la traduction anglaise de ce texte.

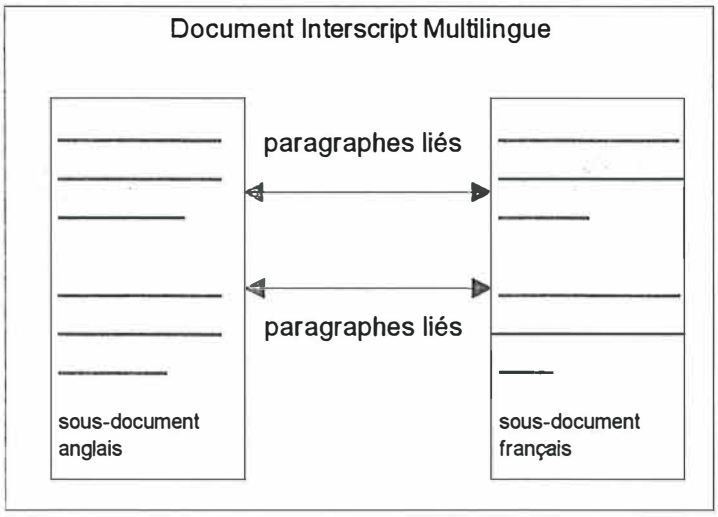


Affichage de sous-documents

- Création de liens entre les sous-documents:

Les sous-documents d'un document multilingue sont liés entre eux. A un paragraphe d'un sous-document correspond un paragraphe d'un autre sous-document.

On liera donc le sous-document français du texte de mode d'emploi de la machine X avec sa traduction, soit le sous-document anglais. A chaque paragraphe du sous-document français correspond un seul paragraphe du sous-document anglais et inversement.

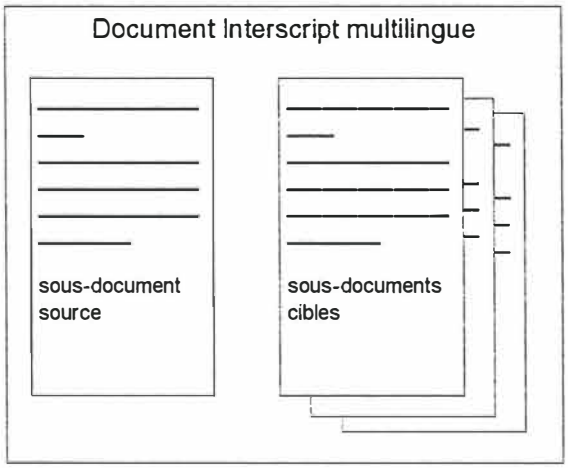


Lien entre les paragraphes de deux sous-documents

- Détermination de sous-documents source et cibles

Un seul sous-document est le sous-document source. Tous les autres sous-documents sont des sous-documents cibles. Le statut de sous-document source est attribué à la création d'un

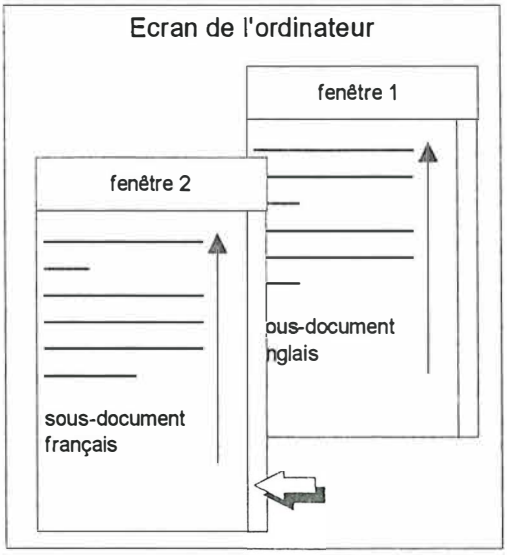
document multilingue. Mais ce statut peut être modifié et on peut ainsi choisir à tout moment un autre sous-document comme étant le sous-document source. Le sous-document source initial devient alors sous-document cible.
Dans notre exemple, le sous-document français est le source et le sous-document anglais est cible. Les autres traductions de ce texte seraient aussi considérées comme cibles.



Sous-document source vs sous-documents cibles

- Défilement parallèle

Etant donné que les paragraphes des différents sous-documents sont liés, le traitement de texte doit offrir un défilement parallèle de deux sous-documents. Si l'utilisateur fait défiler le texte d'un sous-document vers le bas, les textes des autres sous-documents doivent défiler également de manière à voir quel paragraphe anglais dans un sous-document est traduit par quel autre paragraphe du sous-document français.
Si l'utilisateur, qui a affiché à son écran les sous-documents français et anglais fait défiler le texte français vers le haut, le texte anglais dans l'autre fenêtre défilera en même temps en faisant correspondre les paragraphes comme vu précédemment.

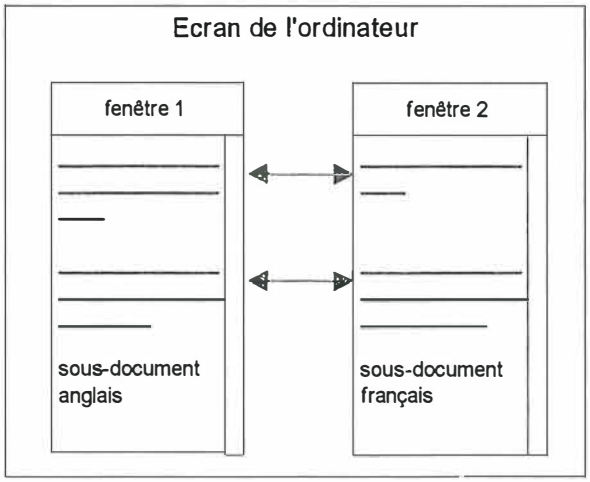


Défilement parallèle de sous-documents

- Mode d'affichage

Le traitement de texte offre deux modes d'affichage des documents multilingues:

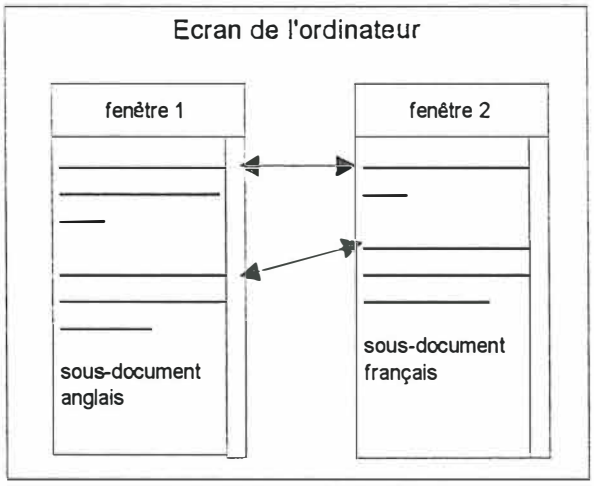
1. Mode 'aligné' : ce mode facilite la comparaison de sous-documents. Des paragraphes traduits dans deux sous-documents sont alignés : le paragraphe d'introduction du sous-document français est au même niveau que le paragraphe d'introduction du sous-document anglais. on perçoit bien de cette façon le lien entre les paragraphes. Même si la traduction anglaise d'un paragraphe est plus longue que le paragraphe français, les paragraphes suivant seront alignés.



Mode d'affichage 'aligné'

2. Mode 'mise en page' : Ici le texte est affiché normalement, en continu. Il a l'aspect qu'il aurait sur une feuille. On ne perçoit pas directement le lien entre les différents paragraphes,

mais on peut avoir un aperçu avant l'impression. On affiche donc le texte français et le texte anglais sans se préoccuper de faire correspondre les différents paragraphes.



Mode d'affichage 'mise en page'

2.2.4.2. Le système de traduction automatique CAT2

Pour une description détaillée de ce système on se référera aux rapports d'activités [CEUSTERS], [DEVILLE_ 1] et [SHARP].
CAT2 est un système de traitement du langage naturel spécialement dédié à la traduction. Il a été créé par l'IAI et constitue le module de traduction automatique utilisé par Apollo.
La traduction des documents d'Apollo peut se faire au niveau du paragraphe, de la phrase, de l'expression ou du mot.
Nous allons essayer de comprendre dans les grandes lignes le fonctionnement de CAT2.

Tout traducteur utilise une grammaire. Une grammaire est un ensemble de déclarations formelles qui définit un ensemble de chaînes de caractères d'un langage et assigne une représentation structurelle à ces chaînes.
La grammaire de CAT2 est un ensemble de règles groupées dans des générateurs et des traducteurs. Les règles des générateurs définissent les structures bien formées à chaque niveau de représentation et celles des traducteurs font correspondre les structures de représentation d'un mot à un niveau avec celles du niveau adjacent. Les générateurs ont donc une fonction de définition, tandis que les traducteurs doivent faire des correspondances entre les différents niveaux de représentation décrits ci-dessous.

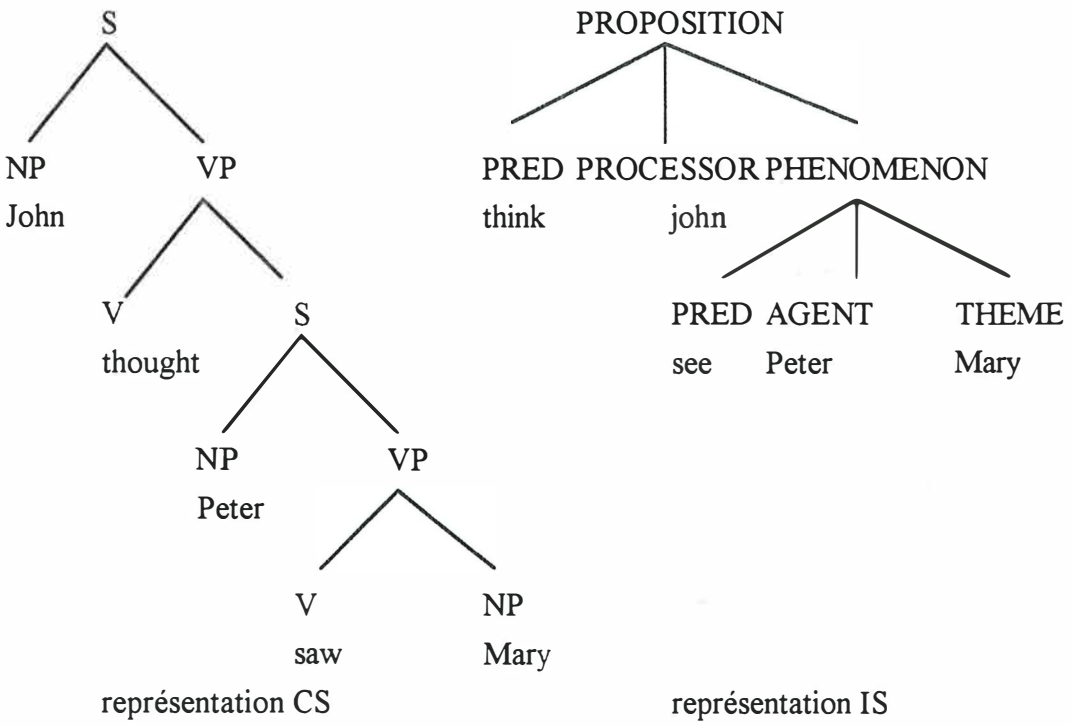
CAT2 divise le monde linguistique en différents niveaux de représentation.

niveau 1: la structure morphologique (MS)
Le générateur morphologique est chargé d'analyser un mot dans un texte ou de générer des mots pendant la synthèse d'un texte. Il en donne une bonne structure morphologique; c'est à dire qu'il définit la structure d'un mot en ses morphèmes. Les morphèmes sont soit libres (mots déterminants, prépositions, adverbes) soit liés : il doivent s'associer à d'autres morphèmes pour former un mot.

niveau 2: la structure syntaxique (CS)
Le niveau syntaxique est utilisé pendant l'analyse grammaticale pour construire une structure de phrase consistante. En input, ce générateur reçoit le résultat de l'analyse morphologique. Le niveau syntaxique autorise les structures créées par le traducteur

niveau 3: la structure de l'interface ou niveau relationnel (IS)
Ce niveau transforme une structure d'un langage en un équivalent dans un autre langage. On se situe ici plutôt du côté de la sémantique.

Prenons la phrase « John thought Peter saw Mary » et donnons sa représentation CS puis IS



représentation CS

représentation IS

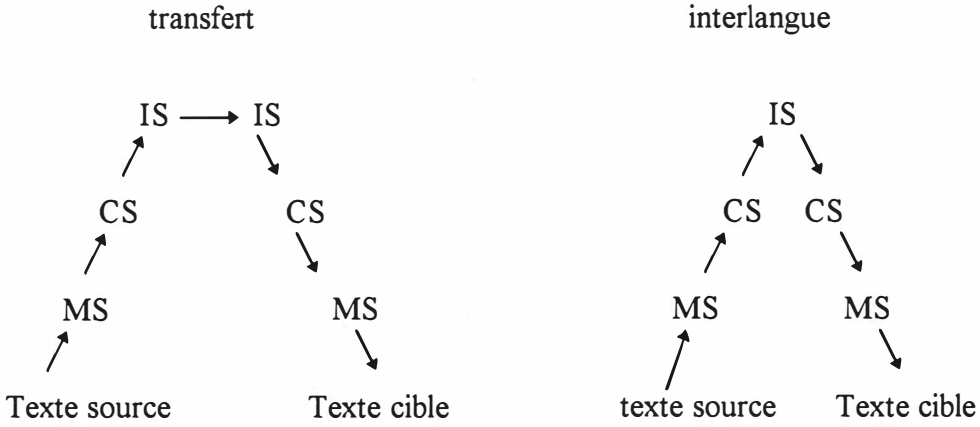
La représentation IS s'éloigne de la syntaxe et permet donc de se rapprocher de la traduction.

La traduction allemande de la phrase étant «Hans dachte, dass Peter Maria sah », la représentation CS diffère totalement tandis que la représentation IS reste la même que pour l'anglais.
Pour traduire une phrase anglaise, il suffit de traduire les éléments de la représentation IS puis de transformer cette représentation en une autre de niveau CS et enfin en une autre encore de niveau en MS.
Le niveau IS permet une traduction simultanée. On observe sur le schéma suivant qu'à partir de l'analyse IS du texte anglais, on construit la représentation IS allemande aussi bien que la française.

MSEN -> CSEN -> ISEN -> ISDE -> CSDE -> MSDE
-> ISFR -> CSFR -> MSFR

Un générateur définira le niveau de représentation CSEN, un autre définira ISEN,... De la même façon, un traducteur spécifiera la transition entre les niveaux CSEN et ISEN et un autre étudiera la transition entre ISEN et ISDE.

CAT2 travaille par approche transfert de la traduction qui est à distinguer de l'approche interlangue.



Le système d'interlangue utilise une langue pivot pour la représentation IS. Cette dernière sera la même pour toutes les langues; elle est indépendante de la langue. Au contraire, le transfert possède une représentation IS par langue.

Pour CAT2, un objet est la représentation d'un mot, expression, phrase sous forme d'un arbre. Chaque noeud de l'arbre contient un ensemble de traits.

CAT2 utilise le mécanisme d'unification de règles. Les règles CAT2 sont en Prolog

Les règles CAT2 décrivent un objet linguistique comme un arbre où les noeuds contiennent des paires d'attributs - valeurs

Il existe différents types de règles utilisées par les générateurs ou les traducteurs.
Les générateurs utilisent les règles 'b' pour construire des structures et des règles 'f' pour affecter des traits. Les traducteurs utilisent les règles 't' pour la traduction de structures d'un niveau à un autre et les règles 'tf' pour traduire le contenu des traits d'une structure entre deux niveaux.

Voyons à présent le processus de traduction de CAT2. On peut remarquer que ce processus passe séquentiellement par les différents niveau de représentation.

Input:
Le processus d'input doit lire le texte au terminal ou dans le fichier et le convertir en objet input prêt à être analysé et traduit. Le texte est divisé en unités de traductions et marques. Par unités de traduction, on entend toute séquence de caractères qui forme une unité cohérente pour la traduction comme par exemple une phrase. Les marques représentent toute chaîne de un ou plusieurs caractères qui constitue un mot, un nombre ou une ponctuation.
Cet objet input est passé à l'analyseur morphologique dans lequel les traits initiaux sont unifiés avec les traits du lexique pour former un objet morphologique.

Analyse morphologique:
Elle prend en input la liste des traits produits dans le processus input et produit pour chaque trait une liste de un ou plusieurs objets atomiques, où la valeur de la chaîne correspond à une ou des entrées lexicales du niveau morphologique correspondant. La liste finale est un objet morphologique et est passée à l'analyseur grammatical pour l'analyse syntaxique.
La reconnaissance des entrées au niveau morphologique est déterminée par la langue source déterminée.

Analyse syntaxique:
Cette analyse prend en input l'objet morphologique et en fait une structure syntaxique sous la forme d'un arbre. Cette analyse est réussie si au moins une structure syntaxique est construite contenant toutes les marques de l'input. Le niveau syntaxique et son correspondant morphologique appartiennent au même langage.
L'analyseur grammatical utilisé à ce niveau travaille en bottom-up, c'est à dire en combinant les marques de l'input en expressions, ces expressions en plus grandes expressions,...

Transformation:

Cette phase transforme le structure d'arbre de l'analyse syntaxique entre différents niveaux de représentation.

Un objet syntaxique peut être transformé en un objet relationnel ou en un autre objet syntaxique. Il en est de même pour l'objet relationnel.

Les objets morphologiques n'entrent pas dans cette phase.

Si on désire transformer un objet source niveau 1 en un objet cible niveau 2, on doit posséder un traducteur liant le niveau 1 au niveau 2 et un générateur de niveau 2. Voici les étapes à exécuter :

1. unifier l'objet source avec le côté source de règle t
2. recommencer pour les sous-objets
3. construire un objet cible provisoire selon les spécifications du côté cible de la règle t en incorporant tous les sous-objets et traduire.
4. localiser une règle b dans le niveau 2 cible qui s'unifie avec l'objet cible provisoire.
5. appliquer toutes les règles tf séquentiellement pour lesquelles le côté source s'unifie avec l'objet source et dont les côtés cibles s'unifie avec l'objet cible provisoire.
6. appliquer les règles f séquentiellement définie au niveau 2 à l'objet cible provisoire. Si toutes les règles f applicables marchent, l'objet provisoire devient définitif et la transformation est terminée.

Synthèse syntaxique

Ici, on transforme une structure input en une structure syntaxique définie par le générateur de niveau syntaxique. La structure input est un objet généré au niveau relationnel ou syntaxique.

Synthèse morphologique

Elle se fait en deux étapes :

1. Isoler les feuilles de l'arbre et donc former une liste de sous-objets atomiques. On aplatit l'arbre.
2. Déterminer la valeur de la chaîne de caractères pour chaque sous-objet au moyen des règles b du niveau morphologique.

La valeur des chaînes de caractères est mise dans une liste, l'objet output.

Output

Prend en input la liste de marques produits par la synthèse morphologique et l'écrit sous forme de texte au terminal ou sur fichier.

CAT2 propose d'autres fonctions :

'Translation path'
Cette fonction donne la séquence des niveaux visités par un objet source pour arriver au niveau cible.

'Translation robust'
Si la traduction échoue à un endroit du processus, le système se met en mode robuste: chaque mot dans l'input est traduit individuellement en utilisant le même processus. Si ça ne marche pas, ça reste comme lors de la traduction.
C'est un mode de fonctionnement qui donne toujours un output.

Estimation de la traduction
CAT2 fait des vérifications et produit un commentaire sur la traduction du texte. Il informe si le texte est traduisible ou non.

Glossaire
CAT2 fournit des glossaires de traduction de textes. L'utilisateur peut choisir un glossaire exhaustif ou sélectif:

- exhaustif: CAT2 produit une liste triée de tous les mots du texte avec les traductions possibles.
- sélectif: Le glossaire ne contient que les mots répondant à certains critères de sélection. Les critères pour Apollo sont des mots avec plusieurs traductions, des mots terminologiques ou des mots non-terminologiques.

2.2.4.3. Le système de gestion de versions SGML

Ce système fournit des fonctions de gestion de versions des documents multilingues. Cette gestion de versions se fait au moyen d'un format SGML. On travaille donc en deux phases:

- on convertit d'abord le document Interscript en document SGML
- ensuite, les différentes versions du document SGML doivent être gérées.

Il s'agit donc d'abord de convertir un document Interscript en un document SGML conforme au type de définition d'un document multilingue Apollo (APOLLO DTD).
On convertit donc un document Interscript en « Apollo DTD document » et inversement. Cette conversion supporte n'importe quel format de caractères ainsi que les tables, images,...

La gestion des versions permet le partage de documents en garantissant sa sécurité d'utilisation et de transformation. On sauve la dernière version du document et toutes les autres versions sont recopiées. Si une ligne d'un document n'a pas été modifiée d'une version à l'autre, elle n'est pas sauvée une deuxième fois. On ne sauve donc que les changements d'une version à l'autre.

Pour Apollo, on se limite à trois fonctions:
- Check out: L'utilisateur peut retrouver une version d'un document identifiée par un numéro de version.
- Check in: L'utilisateur peut sauver une nouvelle version d'un document identifiée par un numéro de version.
- Comparaison: L'utilisateur peut comparer deux versions d'un document. Le résultat de la comparaison est la version la plus récente du document avec des marques sur les paragraphes modifiés par rapport à l'ancienne version.

Ces trois fonctions sont insérées dans l'item « fichier » de la barre de menu du traitement du textes Interscript.

2.2.5. Information des groupes utilisateurs

Cette tâche consiste à informer les utilisateurs de ce que l'on développe pour le projet Apollo. Ceci se fait par des séminaires, une publication des activités et des comptes rendus sur le Word Wide Web,...

2.2.6. Organisation de l'atelier Apollo

Il est également important de vérifier que tous les partenaires du projet sont en ordre administrativement, qu'ils rendent leurs travaux et les comptes rendus s'y rapportant dans les délais prévus,...

2.3. La spécification technique du prototype Apollo

2.3.1. Les exigences des utilisateurs

les utilisateurs ont certaines exigences quant aux outils de traduction qu'ils utilisent :

- ils veulent interagir avec un logiciel convivial tournant sous Win 3.X

- ils voudraient pouvoir afficher leurs documents sous le format de leur traitement de textes préféré.

Voyons si les différents éléments d'Apollo répondent à ces exigences.

- Traitement de textes Interscript

Interscript répond à la première exigence des utilisateurs. En effet, il tourne sous Win 3.X ou Win 95.
Il possède cependant son propre format de document qui ne correspond pas nécessairement à un fichier Word. Mais, au moyen de filtres, un utilisateur peut importer ou exporter des fichiers en format Word sans recourir à un convertisseur. Interscript répond donc également à la deuxième exigence.

- CAT2

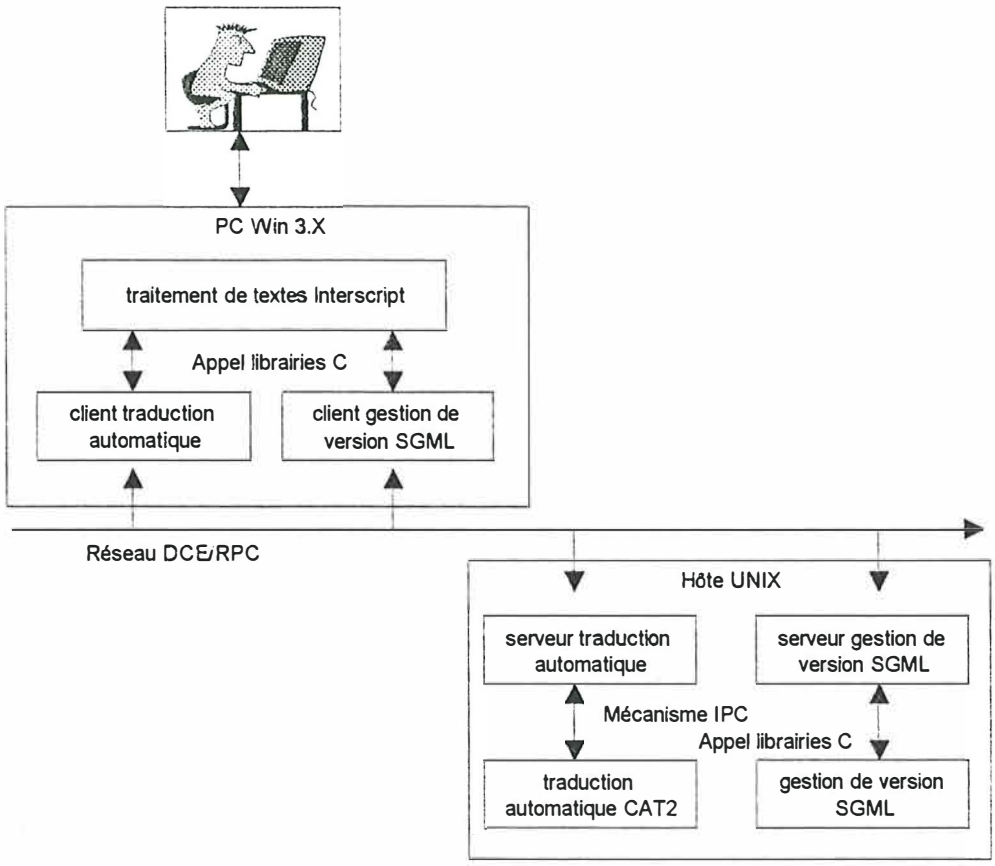
CAT2 a été développé entièrement en Prolog qui n'est disponible que sur une plate-forme UNIX. CAT2 n'est pas disponible en version Win.

- SGML

Cet outil devant être développé depuis le début, il aurait pu être développé en Win. Mais, c'est la version UNIX qui a été choisie. On prévoit en effet une exécution multi-utilisateurs et on a donc besoin d'un OS multi-utilisateurs.

2.3.2. Architecture technique d'Apollo

Les différents conflits techniques que l'on peut rencontrer face aux exigences des utilisateurs sont résolus par le fait que ceux-ci n'interagissent qu'avec Interscript. Et il suffit qu'Interscript puisse accéder aux services des autres modules.



Architecture technique d'Apollo

Interscript interagit avec un client sur un PC. Le PC client communique grâce à un réseau avec un hôte UNIX serveur. Ce serveur interagit avec CAT2 et SGML sur l'hôte UNIX.

L'interaction entre Interscript et le client PC peut être réalisée par appel de librairies C, mécanisme prévu par Interscript.

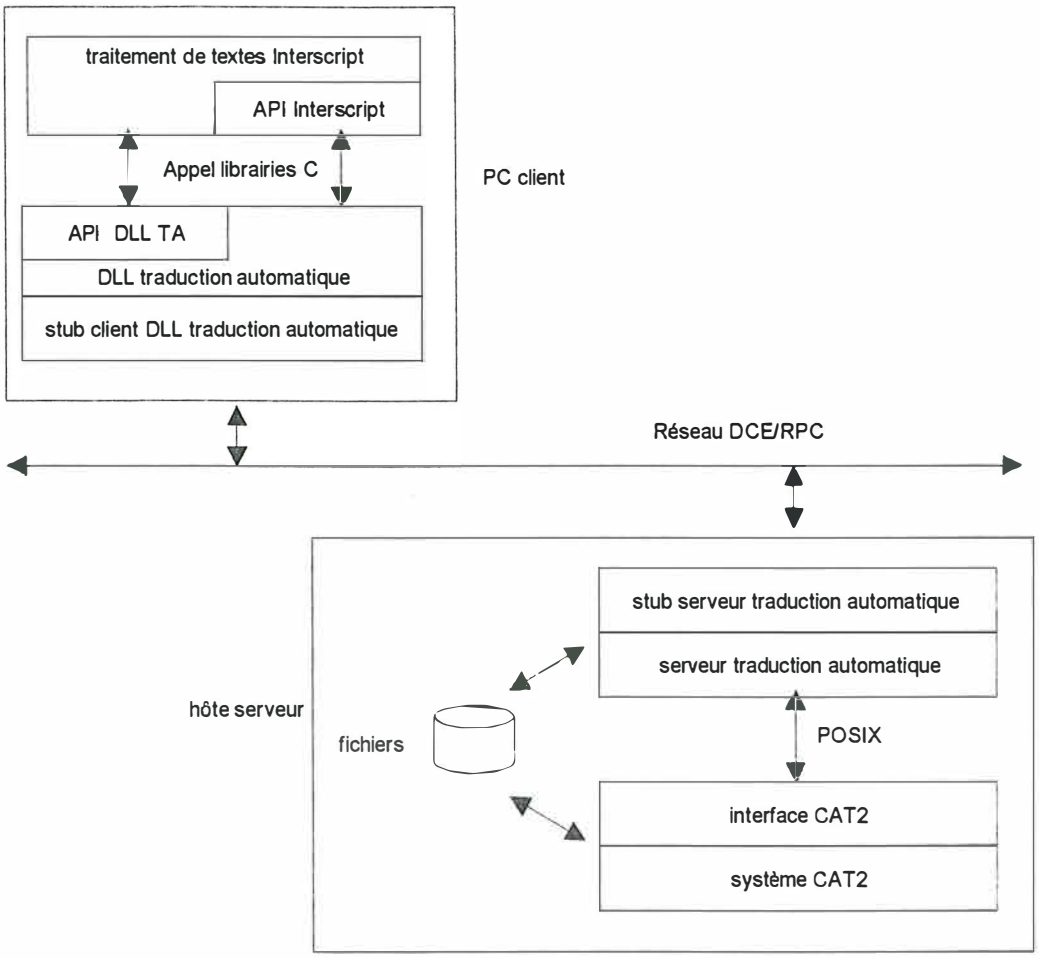
L'interaction entre le serveur de traduction automatique et CAT2 peut se faire par un mécanisme de communication interprocessus standard.

Vu que le module SGML est développé à partir de rien, on choisit de le coupler au serveur par appel à des librairies C.

DCE/RPC est le protocole de communication entre le composant client sur PC et le composant serveur sur UNIX.

Les composants clients sont développés en ANSI C sous Win 3.X. Les composants serveur et SGML sont développé en ANSI C sous UNIX.

2.3.3. L'intégration des fonctionnalités de traduction automatique



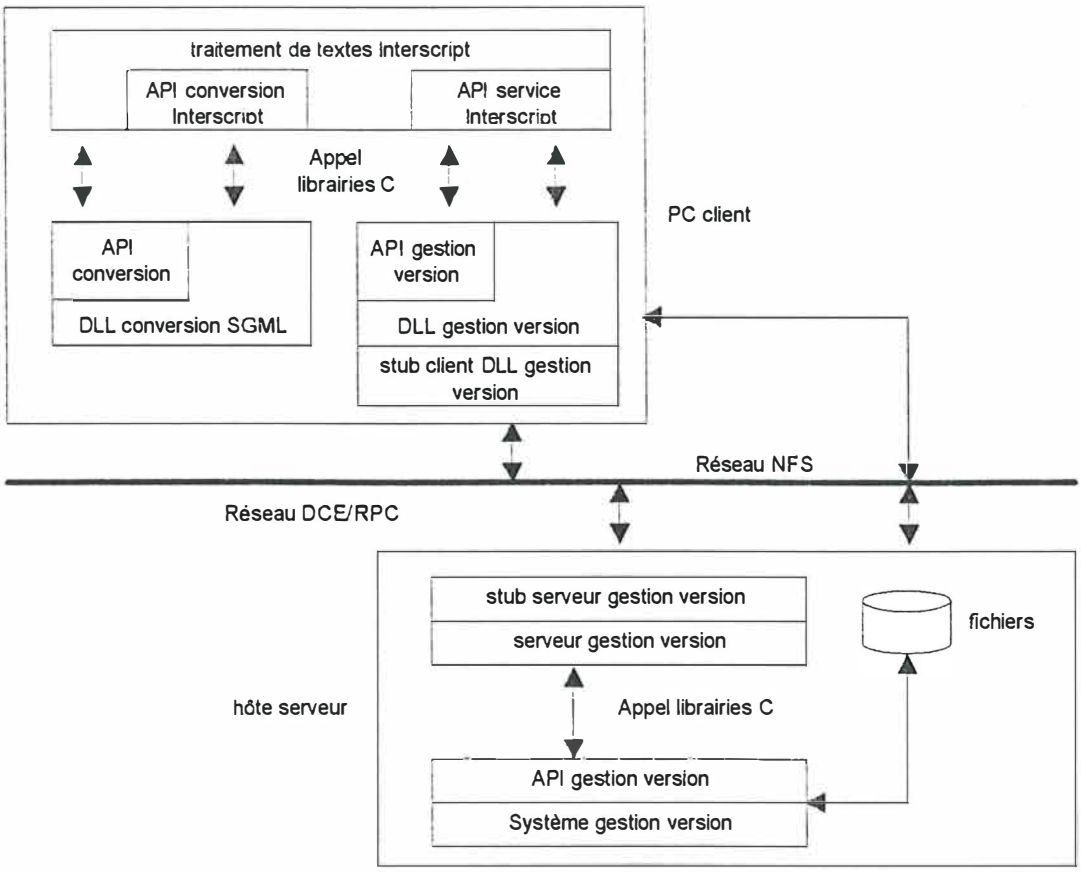
Intégration des fonctionnalités de traduction automatique

Le traitement de textes utilise les services d'une DLL pour offrir à l'utilisateur des fonctions de traduction. Le traitement de textes et les services de la DLL ont chacun une API à travers laquelle ils offrent leurs services. L'API de Interscript contient des fonctions qui donne le corps d'Interscript pour pouvoir étendre son interface. Les API des DLL fournissent des fonctions à Interscript pour pouvoir intégrer les services des DLL dans l'interface Interscript.
La communication entre client et serveur est réalisée par le protocole DCE/RPC
Le serveur de traduction sur UNIX utilise le système CAT2 pour traduire. La communication entre le serveur et CAT2 se fait par POSIX.

2.3.4. Intégration de SGML

J'ai déjà signalé que la gestion des versions se fait en deux phases:

- Un document Interscript doit être convertit en document SGML
- Les différentes versions du document SGML doivent être gérée.



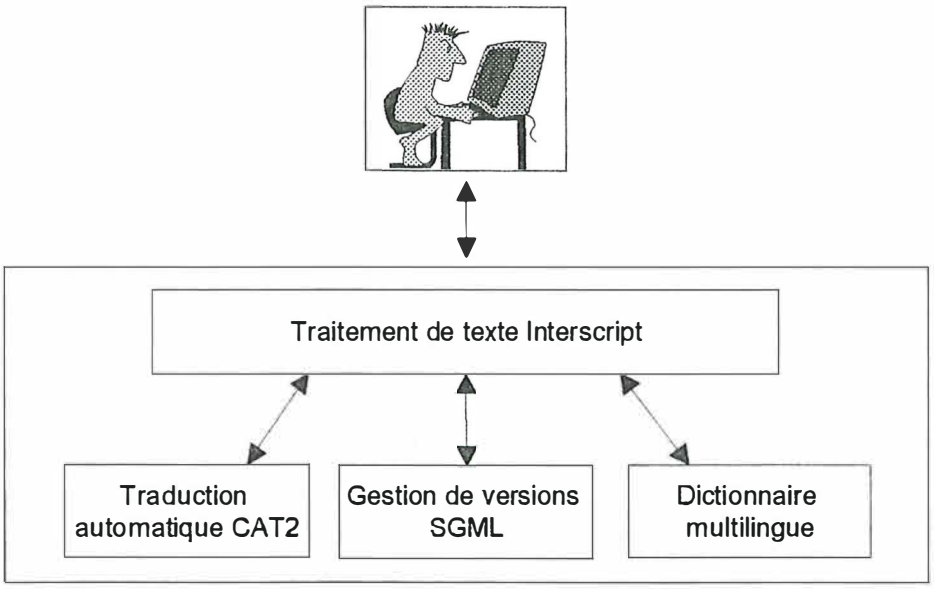
Intégration des fonctionnalités SGML de gestion de version

Interscript utilise une DLL de conversion pour offrir les services de conversion SGML à l'utilisateur. Interscript et la DLL de conversion ont chacune leur API.
En plus des fonctions d'extension de l'interface, l' API Interscript offre des fonctions d'écriture de documents Interscript sans en connaître la représentation interne.
L'API de la DLL fournit des fonctionnalités à Interscript qui lui permettront de l'intégrer à son interface.
La communication entre le traitement de textes et la DLL de conversion se fait par appel de libraires.

Interscript utilise les services d'une autre DLL pour offrir la gestion des versions SGML à l'utilisateur. Cette DLL a aussi une API par laquelle elle offre ses services.
L'API de la DLL a des fonctionnalités pour permettre son intégration dans Interscript.
La communication entre Interscript et la DLL de gestion de version se fait par appel de librairies.
Cette DLL de gestion de version tourne sous UNIX. La communication entre le client et le serveur se fait par le protocole DCE/RPC.
Le partage des fichiers entre le client et le serveur se fait par NFS.

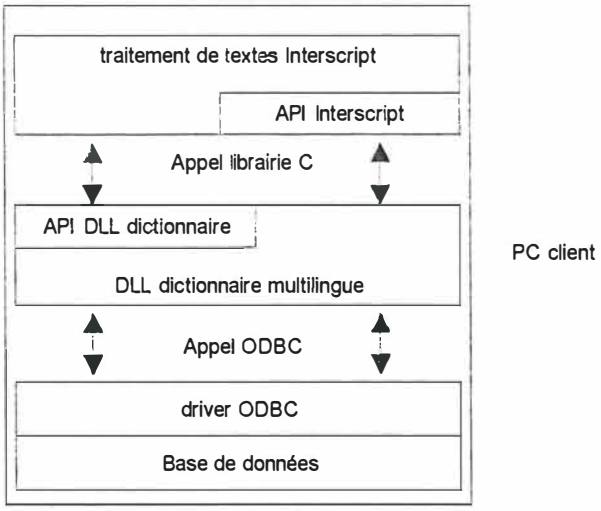
3. COMMENT LE DICTIONNAIRE MULTILINGUE S'INSERE-T-IL DANS LE PROJET APOLLO?

Cette section ne veut pas décrire en détail les fonctionnalités du dictionnaire multilingue. C'est le but de chapitre suivant. Ici, on veut simplement montrer comment ce dictionnaire a été intégré au projet. Le module correspondant au dictionnaire multilingue se rajoute à l'architecture du projet.



Architecture d'Apollo

Pour accéder aux services du dictionnaire multilingue l'utilisateur n'interagit toujours qu'avec Interscript. Un item de menu à été ajoutée au menu spécial qui permet l'accès à ces fonctions.



Intégration du dictionnaire multilingue

La DLL du dictionnaire multilingue a été développée sur PC.
Interscript utilise la DLL du dictionnaire multilingue pour offrir ces services à l'utilisateur. Interscript et la DLL du dictionnaire ont chacune leur API.
L'API de la DLL fournit des fonctionnalités à Interscript qui lui permettront de l'intégrer à son interface.
La communication entre le traitement de textes et la DLL du dictionnaire multilingue se fait par appel de libraires.
Le dictionnaire multilingue accède à une base de données Access, où sont stockés les lexiques pour répondre aux requêtes des utilisateurs. Il y accède au moyen de ODBC. ODBC permettant un accès à n'importe quel type de base de données, on pourra sans problème remplacer la base de données Access par une base de données Oracle si le besoin s'en fait sentir.

Le chapitre suivant va expliciter les fonctionnalités du dictionnaire multilingue ainsi que son implémentation.

Chapitre 4 : L'implémentation du dictionnaire multilingue

1. INTRODUCTION

Ce chapitre décrit le dictionnaire multilingue inclus dans le projet Apollo. On expliquera d'abord les raisons de la conception du dictionnaire ainsi que ses fonctionnalités. Ensuite on étudiera son implémentation. On verra ainsi l'interface du dictionnaire et la base de données dans laquelle il puise les réponses aux requêtes de l'utilisateur.

Ce dictionnaire a été programmé en langage C sous P.C. Il s'agit donc d'une DLL qui est appelée par le traitement de textes Interscript. En effet, l'interface du dictionnaire a été intégrée dans Interscript. Un item a été rajoutée au menu « Spécial » de la barre de menu Interscript. Le lexique utilisé par le dictionnaire multilingue se trouve dans des bases de données Access. Ces dernières sont accédées par des requêtes ODBC.
Tous ces composants seront détaillés dans ce chapitre à l'aide des rapports d'activités rédigés par les partenaires du projet.

2. LES FONCTIONNALITES DU DICTIONNAIRE MULTILINGUE

Rappelons que le but du projet Apollo est de fournir à l'utilisateur des documents multilingues et de gérer ces derniers.
Dans ce cadre, le dictionnaire multilingue est un dictionnaire de consultation qui permet à l'utilisateur d'Apollo de vérifier la traduction d'un mot dans un document multilingue ou de trouver une meilleure traduction de ce mot par rapport au sens qu'il désire lui donner. Prenons l'exemple du mot anglais « papier » que le traducteur automatique traduit en français par « papier ». L'utilisateur consultera le dictionnaire multilingue pour trouver une meilleure traduction française à ce mot, par exemple « article ». Si l'utilisateur est devant un texte qu'il désire traduire, il pourra se référer au dictionnaire pour trouver la bonne traduction d'un terme. En effet le dictionnaire lui fournira des synonymes et les différents sens d'un mot.

Remarquons que contrairement au module de traduction automatique et de gestion de versions, le dictionnaire multilingue traite l'anglais, l'allemand, le français et le néerlandais.

Voici les fonctionnalités que devrait respecter le dictionnaire multilingue.

- L'utilisateur spécifie la langue du terme à traduire.
- L'utilisateur spécifie la langue cible de la traduction.
- Il spécifie le terme pour lequel il désire recevoir de l'information. Ceci se fera soit de manière manuelle en laissant l'utilisateur taper son terme, soit de manière automatique en sélectionnant le terme dans un document.
- L'information consistera en :
  - pour un substantif : genre et pluriel pour les quatre langues
  - pour un adjectif : forme déclinée pour le français et le néerlandais
  - la traduction du terme dans la langue cible.
  - pour toutes les catégories syntaxiques : un exemple d'utilisation du terme en anglais et français
  - pour toutes les catégories syntaxiques : une définition du terme en néerlandais.

D'autres fonctionnalités du dictionnaire ont été imaginées, comme par exemple de lier un adjectif à la définition du substantif lui correspondant. Etant donné que le lexique de départ ne contient que très peu d'adjectifs, cette fonctionnalité ne sera traitée que dans le banc de travail complet du projet, soit Apollo2.

Face à ces spécifications de départ, et étant donné les données contenues dans le lexique, l'implémentation des fonctionnalités du dictionnaire multilingue a été limitée à celles décrites dans l'exemple suivant.
Pour le comprendre il peut être utile de se munir du schéma de la boîte de dialogue du dictionnaire multilingue représenté à la page 71. Les nombres entre parenthèses font référence aux différents objets de cette boîte de dialogue.

Supposons qu'un utilisateur néerlandophone désire traduire en néerlandais le mot français « action » se trouvant dans un document multilingue.

La première possibilité est de sélectionner le mot « action » dans le document Interscript. Ensuite, il suffit de sélectionner « Dictionnaire multilingue » dans le menu « Spécial » d'Interscript pour voir apparaître la boîte de dialogue se rapportant à ce dictionnaire. Cette fenêtre va permettre à l'utilisateur de préciser les caractéristiques des traductions qu'il désire voir apparaître.
Le mot « action » sélectionné par l'utilisateur apparaît maintenant dans la boîte de dialogue remplissant le champ « mot à traduire » (1).

La deuxième solution est de sélectionner directement « Dictionnaire multilingue » dans le menu « Spécial » et de taper le mot « action » dans le champ prévu à cet effet (1).

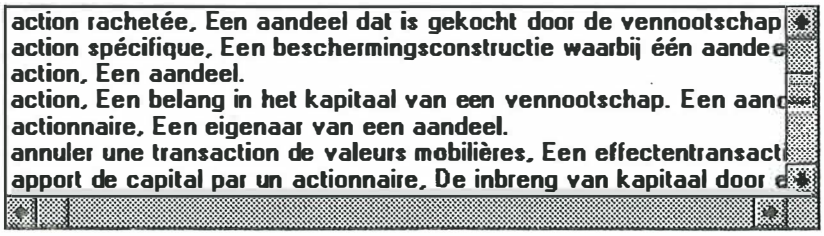
Notre utilisateur doit maintenant sélectionner la langue source de ce mot. C'est une liste de sélection (2) qui lui permettra de choisir le français, l'anglais, l'allemand ou le néerlandais. Etant donné qu'il désire traduire du français au néerlandais, il choisit « français ».

Il faut maintenant répéter cette opération pour choisir la langue cible. L'utilisateur sélectionne donc « néerlandais » dans la liste de sélection correspondant à la langue cible (2).

Lorsque les langues source et cible sont choisies, l'utilisateur peut déterminer si il veut la définition du mot « action » en langue source ou en langue cible. De cette manière, les différents sens du mot que l'on veut traduire seront affichés dans la langue source ou dans la langue cible. Etant néerlandophone, l'utilisateur préfère avoir la définition du mot en néerlandais. il coche donc le bouton radio « langue source » (3).

Lorsque tous les éléments sont spécifiés, il suffit de cliquer sur le bouton « informations » (4) ou bien d'appuyer sur « enter ».
Ceci lance la recherche des informations concernant le mot « action » dans la base de données. Le résultat de cette recherche est une liste de mots ou expressions françaises contenant la suite de caractères « action » suivies de leur définition en néerlandais (5).

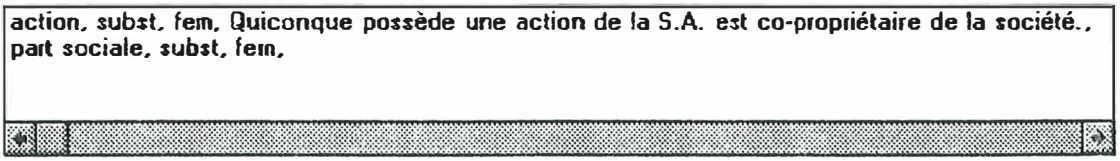
Pour le terme « action » le résultat de la recherche sera par exemple:



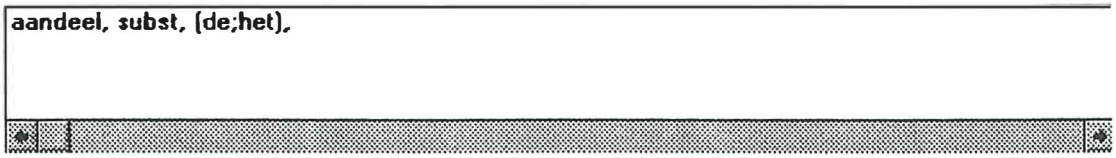
Il faut maintenant sélectionner le mot pour lequel l'utilisateur désire avoir une traduction. Il sélectionne « action » dans la liste de sélection (5).
Cette sélection enclenche l'affichage de la définition en entier du mot « action » dans le champ prévu à cet effet (6) et lance une nouvelle recherche des traductions correspondantes à ce mot dans la base de données.

A la fin de la recherche, on voit apparaître le mot à traduire et ses synonymes dans une liste de sélection (7) et les différentes traductions possibles du mot dans une autre liste de sélection (8).

Dans notre exemple, on retrouve dans la liste des synonymes:



Et comme traduction on a



On voit que chaque terme est suivi d'une série d'informations telles que la catégorie syntaxique, le genre, parfois un commentaire sur la traduction ou une restriction de l'usage du terme ainsi qu'un exemple. Le commentaire et la restriction sont en néerlandais. Un commentaire comme « niet opnemen » signifie par exemple que cette traduction ne s'utilise pas comme telle dans une phrase. La restriction peut par exemple informer l'utilisateur que le terme « Anteil » ne s'utilise que pour certains types de sociétés.

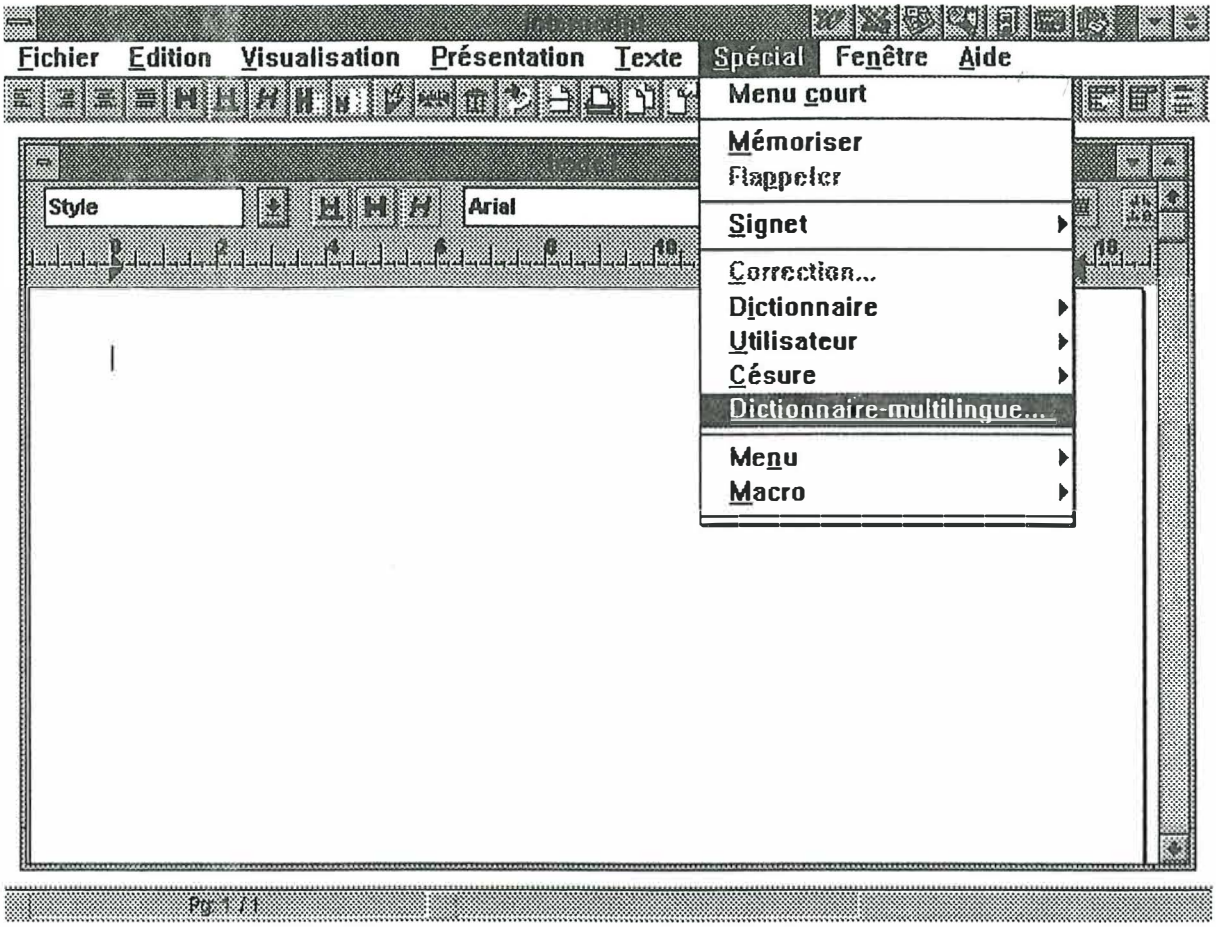
Il est possible de remplacer le mot sélectionné au départ dans le document Interscript par une traduction ou un synonyme. Ainsi, si on désire remplacer le mot « action » dans le document multilingue par un synonyme, il suffit de sélectionner ce synonyme dans la liste de sélection correspondante (7) et lorsque l'on referme la boîte de dialogue « dictionnaire multilingue » en cliquant sur le bouton « quitter » (9), le terme action est remplacé par la dernière sélection.

Par cet exemple, on montre que l'utilisateur du dictionnaire peut s'en servir comme une aide lors de traductions mais aussi comme un vérificateur de traductions. En effet, s'il reçoit un texte traduit et que certains mots ne semblent pas avoir le sens attendu, il peut rechercher les autres sens de ce terme et éventuellement le modifier à l'aide du dictionnaire multilingue.

3. L'INTERFACE DU DICTIONNAIRE MULTILINGUE

L'interface du dictionnaire multilingue a été établie sur base de règles ergonomiques énoncées au cours d'Interface Homme-Machine de F. Bodart.
Le projet Apollo a choisi Interscript comme traitement de textes pour proposer ses fonctionnalités.

Le dictionnaire multilingue est donc intégré dans ce traitement de textes et fait partie de cette interface.
Interscript propose des fonctions qui permettent d'ajouter un item à un menu dans sa barre de menu. Dans ce cas-ci, on a intégré l'item « dictionnaire multilingue ... » au menu « Spécial » dans lequel on retrouve déjà d'autres dictionnaires.



Barre de menu d'Interscript

Le signe « ... » dans le nom de l'item signifie que la sélection de cet item engendre l'affichage d'une boîte de dialogue. Ici c'est la boîte de dialogue « spécial Dictionnaire Multilingue » qui apparaît.

Nous avons essayé d'afficher le plus d'informations possibles sur une seule fenêtre pour éviter à l'utilisateur de devoir cliquer sur toute une série de boutons.

Le champ « mot à traduire » (1) doit contenir une seule valeur qui est au départ inconnue par le logiciel. Les règles ergonomiques suggèrent alors de saisir cette information donnée par l'utilisateur dans un simple champ d'édition. (1)

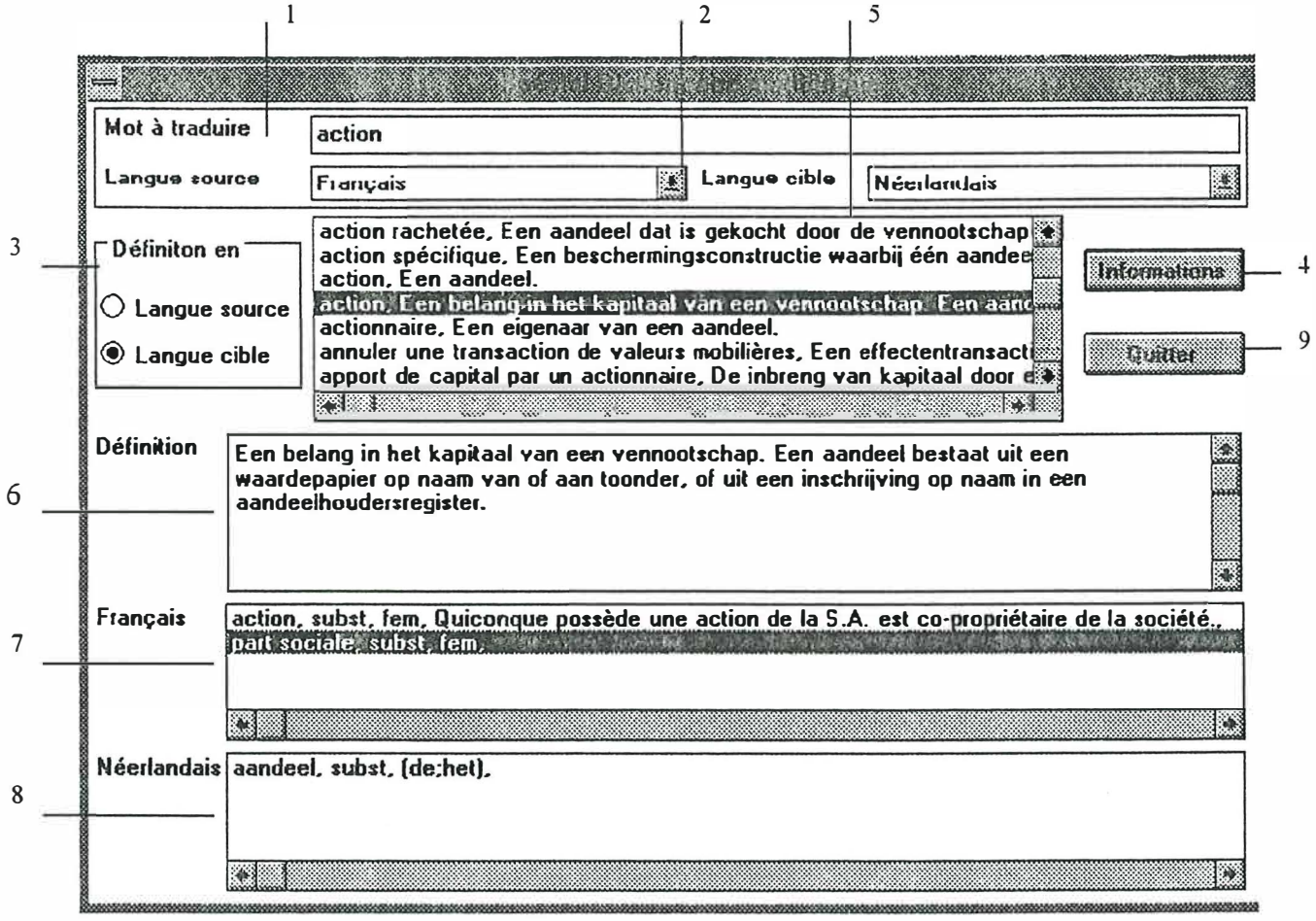
Par contre, pour le choix de langue source, le domaine de sélection est connu puisque l'utilisateur ne peut choisir qu'entre quatre valeurs prédéfinies. On a donc choisi d'afficher les quatre langues dans une liste de sélection. Et il en est de même pour la liste de sélection de la langue cible. (2)

Lorsque l'utilisateur choisit la langue dans laquelle il veut avoir la définition des termes, il a deux possibilités prédéfinies. Lorsque le domaine de sélection est connu et que le nombre de valeurs possibles est compris entre deux et trois, la règle ergonomique propose d'afficher les différents choix sous forme de boutons radio. (3)

Les différents boutons apparaissant dans une boîte de dialogue doivent être affichés soit en colonne à droite, soit en ligne en dessous dans la boîte de dialogue. (4) et (9)

Les listes de sélection (5), (7) et (8) ont été choisies pour les mêmes raisons que celles énoncées pour les listes de sélection (2).

Le champ contenant la définition (6) aurait pu être un simple champ d'édition puisqu'il ne contient qu'une valeur. Mais cette valeur est parfois de grande taille et on a rajouté une barre de défilement verticale pour pouvoir la lire en entier.



Boîte de dialogue « Spécial dictionnaire multilingue »

Comme nous l'avons vu dans l'exemple des pages 67 et suivantes les différents champs de cette boîte de dialogue s'affichent au fur et à mesure que la requête se précise.
En effet, l'utilisateur remplit d'abord les champs correspondant au mot à traduire, à la langue source, à la langue cible et à la langue de la définition.
Le champ du mot à traduire (1) peut contenir un texte libre. Il a une longueur maximale de 255 caractères.
Les champs correspondant aux langues source et cible (2) sont des listes de sélection dans lesquelles on ne peut sélectionner qu'une seule langue, soit l'anglais, l'allemand, le français ou le néerlandais. L'utilisateur ne peut pas rajouter des items à ces listes.
L'utilisateur détermine en quelle langue il veut voir s'afficher la définition des mots par une sélection d'un des boutons radio (3).
Le bouton « Information » (4) sert à lancer la recherche dans la base de données.
Le résultat de cette recherche amène l'affichage dans la liste de sélection (5) de tous les concepts trouvés dans la base de données qui contiennent la chaîne de caractères introduite par l'utilisateur dans le champ « mot à traduire » (1). Chaque concept est suivi de sa définition dans la langue choisie par l'utilisateur. L'utilisateur ne peut rien changer dans cette liste. La seule action qu'il peut y faire est de sélectionner un item.
Un clique sur un des items de la liste de sélection (5) entraîne l'affichage de la définition en entier dans le champs « définition » (6). Ce champ n'est pas accessible en écriture à l'utilisateur. Il est impossible d'en modifier le contenu à partir de l'interface. C'est un champ limité à la lecture.
En plus de l'affichage de la définition, on voit apparaître dans la liste de sélection (7) le mot sélectionné dans la liste de sélection (5) suivi de certaines informations grammaticales liées à son usage ainsi que des synonymes de ce mot suivis également de certaines informations.
Simultanément, la liste de sélection (8) est complétée des traductions possibles du mot choisi dans la liste de sélection (5). Tout comme les synonymes, les différentes traductions sont suivies d'informations grammaticales.
Les listes de sélection (7) et (8) ne peuvent pas être modifiées par l'utilisateur. Le seul usage que celui-ci peut en faire est de cliquer sur un des items de ces deux listes, ce qui entraînera l'affichage du mot sélectionné dans le document Interscript.

4. LES BASES DE DONNEES

Comme je l'ai déjà signalé, les lexiques utilisés par le dictionnaire multilingue sont stockés dans des bases de données Access.

Nous avons vu que ces lexiques sont des ressources de Spectrum et de ABB qui ont été retravaillées. Elles sont maintenant regroupées dans des fichiers qui ont la même structure à deux ou trois champs près. On se référera au chapitre 1 pour plus de précisions sur le contenu de ces fichiers.

Pour accéder à des données, le dictionnaire a besoin de bases de données. Le contenu des fichiers Spectrum et ABB ont donc été insérés dans deux bases de données différentes, accessibles toutes deux par le dictionnaire multilingue moyennant une configuration ODBC.

4.1. Le schéma conceptuel des bases de données

On a voulu élaborer un schéma conceptuel de base de données qui convient non seulement au projet Apollo mais à n'importe quel autre dictionnaire multilingue.
La question à se poser est la suivante : Quelles informations désire-t-on recevoir lors de l'utilisation d'un dictionnaire multilingue ? Des dictionnaires papiers ainsi que d'autres dictionnaires électroniques ont été consultés afin d'y répondre.
Etant donné les données qui nous ont été fournies et les besoins de ce type de dictionnaire, voici le schéma qui a été retenu.

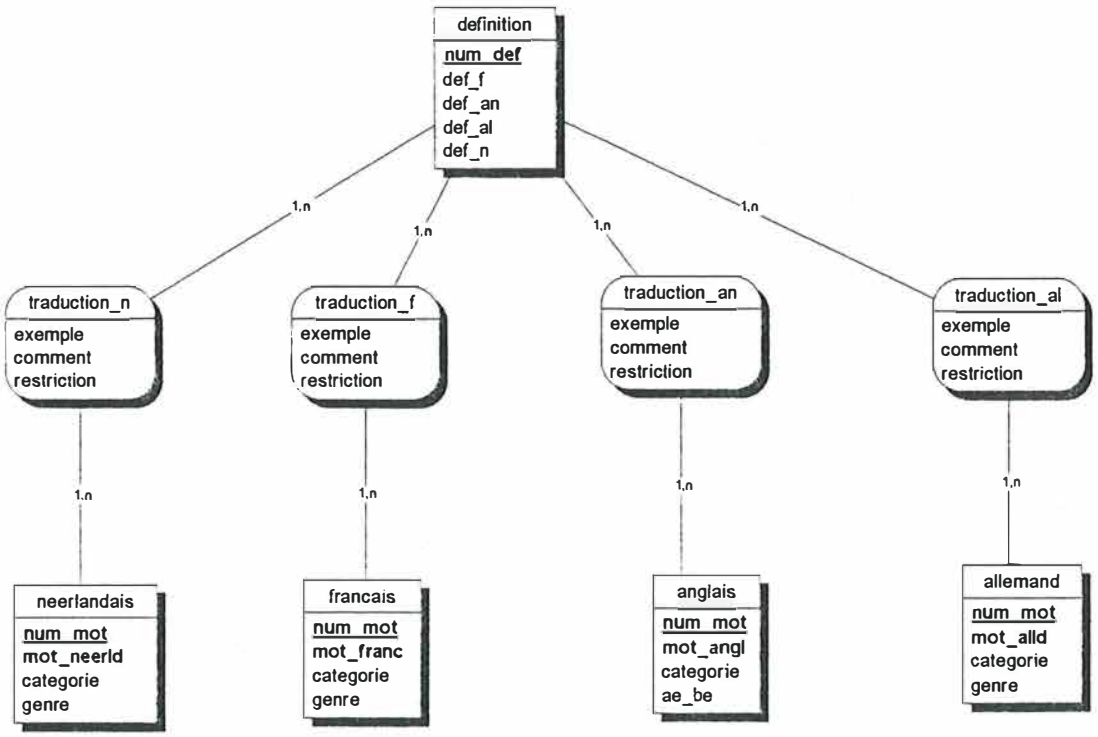


Schéma conceptuel utilisé pour Apollo

Comme le montre ce schéma, à chaque définition correspond au moins un mot en une langue. Et une définition peut correspondre à plusieurs mots en une langue. Il s'agit alors de synonymes. Chaque mot d'un langage possède au moins un numéro de définition qui lui donne son sens. Un mot peut avoir plusieurs sens et donc plusieurs définitions.
« num_def » a été choisi comme identifiant de la table définition car les autres champs ne sont pas obligatoires.
Les champs « exemple », « comment » et « restriction » sont placé dans la relation car leur valeur dépend du sens d'un mot.
« num_mot » est identifiant des tables concernant les langues car ces tables peuvent contenir deux fois le même mot ayant des sens différents. Par exemple, la table anglais pourra contenir une première fois le mot « fall » dans le sens « automne » et une deuxième fois dans le sens « tomber ».

le schéma conceptuel a donné la structure des bases de données Access. Ces bases de données comprennent neuf tables.

Voici le schéma relationnel logique déduit du schéma précédent :

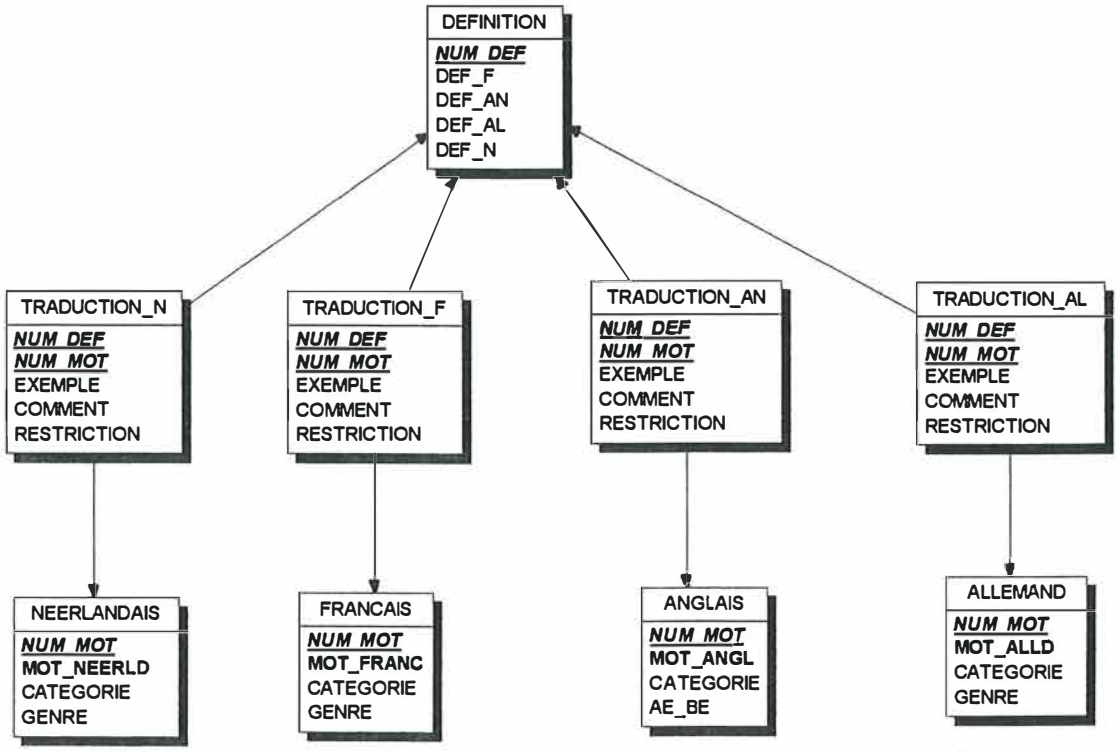


Schéma relationnel logique utilisé pour Apollo

4.2. La structure de la base de données

Détaillons maintenant les différentes tables.

La table « définition » :

- **num_def** est la clé primaire de la table. Il identifie un ensemble de définitions c'est à dire une définition dans chaque langue.
- **def_an** contient la définition en anglais
- **def_f** contient la définition en français
- **def_n** contient la définition en néerlandais
- **def_al** contient la définition en allemand

Remarquons que nos lexiques ne nous ont fourni que des définitions en néerlandais. Cependant si l'on possède un lexique plus complet ou si on décide de traduire la définition que l'on possède en d'autres langues, les autres champs sont utiles.

La table « anglais » :

- **num_mot** est la clé primaire de la table. Elle identifie un mot
- **mot_angl** contient le mot anglais
- **categorie** contient la catégorie syntaxique du mot anglais
- **ae_be** est un champ qui contient un commentaire sur l'utilisation de la traduction anglaise. Il précise si le terme est anglais ou américain.

La table « traduction_an » :

Cette table fait le lien entre un mot et ses différents sens.

- **num_def** et **num_mot** forment la clé primaire de cette table. Ce sont les clés d'accès aux tables « définition » et « anglais ».
- **exemple** contient un exemple qui illustre l'utilisation du mot ayant un certain sens.
- **comment** contient un commentaire sur la traduction du mot utilisé dans un certain sens.
- **restriction** donne une restriction sur l'utilisation de ce mot dans un certain sens.

Le commentaire et la restriction sur les mots se trouvent au niveau des relations entre la table contenant les définitions et les tables contenant les mots. En effet, ces commentaires et restrictions se rapportent à un certain sens du mot et donc à une certaine définition.
La base de données ne contient que quelques exemples. Cependant il est possible d'en ajouter pour chaque terme. Les exemples que l'on possède pour le moment ont été inscrits manuellement dans la base de données.

La table « français » :

- **num_mot** est la clé primaire de cette table. Il identifie un mot.
- **mot_franc** contient un mot français.
- **categorie** contient la catégorie syntaxique du mot français.
- **genre** contient le genre du mot français

La table « traduction_f » :

- **num_def** et **num_mot** forment la clé primaire de cette table. Elles sont aussi les clés d'accès aux tables « définition » et « français ».
- **exemple** fournit un exemple illustrant l'utilisation du mot français ayant un certain sens.
- **comment** contient un commentaire sur la traduction du mot utilisé dans un certain sens.
- **restriction** donne une restriction sur l'utilisation de ce mot dans un certain sens.

La table « néerlandais » :

- **num_mot** est la clé primaire de cette table. Il identifie un mot.
- **mot_neerld** contient un mot néerlandais.
- **categorie** contient la catégorie syntaxique du mot néerlandais.
- **genre** contient le genre du mot néerlandais.

La table « traduction_n » :

- **num_def** et **num_mot** sont clé primaire de cette table. Elles sont également les clés d'accès aux tables « définition » et « néerlandais ».
- **exemple** fournit un exemple illustrant l'utilisation du mot néerlandais ayant un certain sens.
- **comment** contient un commentaire sur la traduction du mot utilisé dans un certain sens.
- **restriction** donne une restriction sur l'utilisation de ce mot dans un certain sens.

La table « allemand » :

- **num_mot** est la clé primaire de cette table. Il identifie un mot.
- **mot_alld** contient un mot allemand.
- **categorie** contient la catégorie syntaxique du mot allemand.
- **genre** contient le genre du mot allemand.

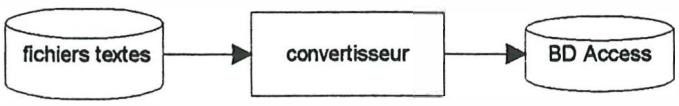
La table « traduction_al » :

- **num_def** et **num_mot** forment la clé primaire de cette table. Ils sont également les clés d'accès aux tables « définition » et « allemand ».
- **exemple** fournit un exemple illustrant l'utilisation du mot allemand ayant un certain sens.
- **comment** contient un commentaire sur la traduction du mot utilisé dans un certain sens.
- **restriction** donne une restriction sur l'utilisation de ce mot dans un certain sens.

On peut remarquer une certaine homogénéité des tables. Tous les champs des tables ne seront pas remplis avec l'information fournies par Spectrum ou ABB, mais il suffirait d'utiliser un lexique plus complet pour que tous les champs aient une valeur.

4.3 Le contenu des bases de données

J'ai donc élaboré deux bases de données, l'une contenant les données Spectrum et l'autre contenant les données ABB. Ces bases de données ont été remplies automatiquement par un petit programme. Ce programme prend en entrée les fichiers Spectrum et ABB ainsi que deux bases de données vierges. En sortie, il fournit les bases de données remplies avec les données l'une de Spectrum, l'autre d'ABB.



Pour remplir les bases de données, il a fallut lire chaque caractère des fichiers textes et en former des données.

L'insertion des données dans les champs de la base de données se fait par des requêtes ODBC.

Voici l'explication des fonctions du convertisseur. Pour une meilleure compréhension de ce qui suit, il est conseillé de reprendre la structure des fichiers Spectrum et ABB décrite dans la premier chapitre ainsi que le code source en annexe 1.

1. InitialiseBd

Cette fonction:

- ouvre le fichier Spectrum ou ABB en lecture
- se connecte à la base de données
- lit le fichier caractère par caractère jusqu'à ce qu'elle rencontre la fin de fichier
- se déconnecte de la BD
- ferme le fichier une fois la fin de fichier rencontrée.

La lecture du fichier se fait en 5 étapes:

1. Lecture de la définition :

Le programme lit la définition dans le fichier. Contrairement au fichier Spectrum, le fichier ABB n'en contient pas. Il faut donc tester si la définition existe avant de l'insérer dans la table « définition ». Etant donné que toutes les définitions que l'on possède sont en néerlandais, elles sont insérées dans le champ « def_n » de cette table.
Que le champ contenant la définition soit vide ou non, on inscrit un numéro de définition dans le champ « num_def » de cette même table.

2. Lecture des éléments anglais:

Nous avons vu que les fichiers textes contiennent au plus quatre traductions dans chaque langue. Il faut donc boucler quatre fois dans ce processus de lecture des éléments anglais pour lire les éléments se rapportant aux quatre traductions.
Le programme lit tous les éléments se rapportant à la traduction anglaise du terme avant de les insérer dans les tables « anglais » et « traduction_an ».
Il lit le premier mot anglais. Il se peut que ce dernier n'existe pas. Il est alors marqué par ce signe : « - ».
Si le programme rencontre ce signe, il ne lit pas les autres éléments concernant l'anglais et passe tout de suite aux éléments français.
Si le programme trouve une traduction anglaise, il la stocke dans une variable. Ensuite il continue la lecture de la catégorie syntaxique, de l'élément ae-be, de la restriction, du commentaire et les stocke également dans différentes variables.

La lecture terminée, le programme teste si ce mot ne figure pas déjà dans la table et évite ainsi de le dédoubler. Il faut cependant tenir compte du fait qu'un même mot peut être verbe ou nom. Il doit alors figurer deux fois dans la table. Par exemple le mot anglais « fall » peut signifier « automne » et est alors un nom, ou bien il signifie « tomber » et est traité comme verbe. Il doit donc apparaître deux fois dans la table.
Ces vérifications faites, le programme insère ces éléments dans les différents champs correspondant des tables « anglais » et « traduction_an ».
En résumé, on ne lit donc que si le mot est différent de « - » et on n'insère le mot que s'il est différent de «- » . De plus, on n'insère que si le mot ne figure pas déjà dans la table avec la même catégorie.
Si le mot est égal à « - », on lit tous les champs jusqu'à ce qu'on arrive aux termes français.

3. Lecture des éléments français :

Ici aussi on va boucler pour retrouver toutes les traductions françaises. Le processus est le même que pour l'anglais.
Le programme commence par lire le mot. Selon sa valeur il continue ou stoppe la lecture des éléments français. S'il continue, il lira le genre du mot, sa catégorie syntaxique, la restriction et le commentaire s'y rapportant.
Après lecture de tous les éléments français et la vérification de leur non appartenance à la table, ils sont insérés dans les tables « français » et « traduction_f ».
En résume, on ne lit donc que si le mot est différent de « - ». On n'insère que si le mot est différent de « - » et que si le mot ne figure pas déjà dans la table avec la même catégorie et le même genre.
Si le mot est égal à « - », on lit tous les champs jusqu'à ce qu'on arrive aux termes néerlandais.

4. Lecture des éléments néerlandais :

On suit à nouveau le même processus que pour les langues précédentes.
On va donc lire comme précédemment le mot, puis éventuellement le genre du mot, la catégorie syntaxique et l'élément znl_nnl. Ce dernier élément indique si on utilise ce mot dans le nord des Pays-Bas et sa valeur vaut alors « nnl », ou dans le sud des Pays-Bas et en Belgique auquel cas sa valeur vaut « znl ». Ensuite on lit l'abréviation si elle existe.
Il faut ensuite vérifier si le mot ne figure pas déjà dans la table. Sinon, on insère les éléments dans les champs correspondant des tables « néerld » et « traduction_n ».
Si le programme a trouvé une abréviation, il la considère comme un mot et l'insère dans la table avec la même catégorie et le même genre que le mot qu'il abrège. L'abréviation est considérée comme un synonyme du mot.

L'élément znl_nnl est insérée dans le champ restriction de la table « traduction_n »
En résumé, on ne lit donc que si le mot est différent de « - » et on n'insère que si le mot est différent de « - ». On n'insère que si le mot ne figure pas déjà dans la table avec la même catégorie et
le même genre.
Si on a une abréviation, on l'affiche dans la colonne 'mot' de la table néerlandais.
Si le mot est égal à « - », on lit tous les champs jusqu'à ce qu'on arrive aux termes allemands.

5. Lecture des éléments allemands :

Le traitement de la langue allemande est exactement le même que celui de la langue française. En effet, on commence par lire le mot, puis selon sa valeur on continue à lire le genre, la catégorie syntaxique, la restriction et le commentaire.
Après les tests, on insère les éléments dans les champs des tables « allemand » et « traduction_al ».
En résumé, on ne lit donc que si le mot est différent de « - » et on n'insère que si le mot est différent de « - ». On n'insère que si le mot ne figure pas déjà dans la table avec la même catégorie et le même genre.
Si le mot est égal à « - », on lit tous les champs jusqu'à ce qu'on arrive au champ numéro 100 dans le fichier, pour pouvoir recommencer une série de lecture avec un nouveau concept néerlandais et une nouvelle définition.

2. BuildString

Cette fonction lit une série de caractères jusqu'à ce qu'elle rencontre un « return » et en fait une chaîne de caractères.
Elle lit le fichier texte caractère par caractère.
Elle alloue au départ 256 caractères à un tampon dans lequel elle écrit ce qu'elle lit dans le fichier texte. Si il n'y a plus de place dans le tampon, elle réalloue 256 caractères à ce tampon de façon à ce qu'il puisse contenir la totalité de la chaîne de caractères.

5. LES REQUETES DANS LA BASE DE DONNEES ET L'AFFICHAGE DES RESULTATS DANS L'INTERFACE

Après avoir décrit d'une part l'interface et d'autre part les bases de données utiles au dictionnaire multilingue, on va maintenant étudier le lien entre ces composants.

C'est la DLL du dictionnaire multilingue qui perçoit ce que l'utilisateur lui transmet par l'intermédiaire de l'interface du dictionnaire. Après capture et analyse de ces données, elle accède à la base de données Access par des requêtes ODBC. ODBC a été choisi pour permettre au moment opportun de changer de base de données facilement. A n'importe quel moment, il sera possible d'accéder à un base de données Oracle sans modifier le code de la DLL. Il suffit de préciser au driver ODBC à quelle base de données on veut accéder. On précise de quelle type elle est (Access, Oracle,...), son chemin d'accès et son nom.
En recevant le résultat de la recherche dans la base de données, la DLL affiche les informations dans l'interface.

Le code source de cette DLL se trouve en annexe 2.

Voici une description des fonctions principales de cette DLL que j'ai développé.

1. void SelectionNumerosMotPremier (HWND hDlg)

Fonctionnalité : SelectionNumeroMotPremier

Interface

- arguments : hDlg (identifiant de la boîte de dialogue « spécial dictionnaire multilingue »)
  bufmot (variable contenant le mot à traduire)
- résultats : tNumMot (tableau de numéro de mot)
- exceptions : message (messages d'erreur apparaissant lorsque bufmot (donné par l'utilisateur) n'est pas trouvé dans la table de la base de données ou lorsque la base de données n'a pas la bonne structure et que la fonction ne retrouve pas la table qu'elle doit consulter dans la base de données.)

Règles de traitement

- Précondition :
  la base de données est accessible
  la variable globale « bufmot » est initialisée.
- Postcondition :
  le tableau tNumMot contient tous les numéros de mot répondant à la requête SQL1
- Excepte : message

Requêtes ODBC:

- **SQL1**: cette requête sélectionne les numéros des mots dans lesquels on retrouve la chaîne de caractères entrée par l'utilisateur. Cette sélection doit se faire dans le champ « mot » de la table correspondant à la langue source précisée par l'utilisateur.

Cette fonction sélectionne dans la base de données les numéros des mots qui contiennent la chaîne de caractères entrée par l'utilisateur dans le champ « mot à traduire » (1). Pour ce faire, elle consulte la table de la base de données correspondant à la langue source donnée par l'utilisateur. Elle met les numéros de mot qu'elle trouve dans le tableau « tNumMot ». Elle prévient l'utilisateur de certaines erreurs lorsqu'elles surviennent.

2. void SelectionDefinition (HWND hDlg)

Fonctionnalité : SelectionDefinition

Interface

- arguments : hDlg (identifiant de la boîte de dialogue « spécial dictionnaire multilingue »)
  tNumMot (tableau de numéro de mot)
- résultats : tNumDef (tableau de numéro de mot)
  liste (mot suivi de sa définition à afficher dans la liste de sélection)
- exceptions :

Règles de traitement

- Précondition :
  la base de données est accessible
  le tableau tNumMot est initialisée.
- Postcondition :
  le tableau tNumDef contient tous les numéros de définition répondant à la requête SQL2.
  La liste de sélection IDC_LIST contient les mots, correspondant au numéro de mot de tNumMot, suivi de leur définition correspondant au numéro de définition de tNumDef
  A chaque mot inséré dans la liste IDC_LIST est associé le numéro de définition correspondant.
- Excepte :

Requêtes SQL:

- **SQL2**: Cette requête est exécutée pour chaque numéro de mot contenu dans le tableau « tNumMot ». Elle sélectionne les numéros de définition correspondant au numéros de mots du tableau « tNumMot ». Cette sélection se passe dans la table de traduction correspondant à la langue source déterminée par l'utilisateur.
- **SQL3**: Cette requête est exécutée pour chaque numéro de mot du tableau « tNumMot » et pour chaque numéro de définition trouvé dans la requête SQL2. Elle sélectionne le mot en langue source et la définition dans la langue déterminée par le bouton radio, correspondant

aux numéros de mot et de définition dans la table définition et dans la table correspondant à la langue source.

Cette fonction sélectionne les numéros de définitions correspondant aux numéros de mot trouvés dans la fonction 1. La sélection se fait dans les tables de traduction à partir du tableau « tNumMot » remplit à la fonction 1.
Elle place les numéros de définitions qu'elle trouve dans le tableau « tNumDef ».
Se basant sur ces numéros, elle sélectionne maintenant dans la table « définition » la définition correspondant à chaque numéro de définition trouvé dans la requête précédente.
La définition qu'elle sélectionne est celle écrite dans la langue que l'utilisateur a choisi pour afficher la définition à l'aide des boutons radio.
Ensuite, la fonction affiche le mot suivi de sa définition dans la liste de sélection prévue a cet effet (5).
A chaque chaîne de caractères qu'elle insère dans la liste de sélection, la fonction associe le numéro de définition lui correspondant, de telle manière que lorsque l'utilisateur cliquera sur un item de la liste (5), la fonction obtienne directement le numéro de définition associé au choix de l'utilisateur. Ceci permet de repérer rapidement le sens du mot.

3. void SelectionNumerosMot (BYTE buffer [50], LONG int numero)

Fonctionnalité : SelectionNumeroMot

Interface

- arguments : buffer (variable contenant une langue)
  numero (variable contenant un numéro de définition)
- résultats : tNumMot (tableau de numéro de mot)
- exceptions :

Règles de traitement

- Précondition :
  la base de données est accessible
  la variable « buffer » est initialisée
  la variable « numero » est initialisée
- Postcondition :
  le tableau tNumMot contient tous les numéros de mot répondant à la requête SQL4
- Excepte :

Requêtes SQL :

- **SQL4**: Cette requête sélectionne les numéros de mots correspondant au numéro de définition associé à la sélection de l'utilisateur dans la liste de sélection (5). Cette recherche se fait dans la table de traduction correspondant à la valeur de la variable « buffer ».

Lorsque l'utilisateur sélectionne un item de la liste de sélection (5), la fonction retrouve les numéros de définition correspondant à cette sélection de l'utilisateur. Ensuite, sur base de ce numéro, elle sélectionne les numéros de mot qui y sont associés dans les tables de traduction.
Cette fonction est appelée pour retrouver les numéros de mots correspondant aux synonymes du mot sélectionné par l'utilisateur dans la liste de sélection (5) et pour retrouver les numéros de mots correspondant aux traductions de ce même mot.
Elle met les numéros qu'elle trouve dans le tableau « tNumMot ».

4. void SelectionSynonymes (HWND hDlg, LONG int def)

Fonctionnalité : SelectionSynonymes

Interface

- arguments : hDlg (identifiant de la boîte de dialogue « spécial dictionnaire multilingue »)
  def (variable contenant un numéro de définition)
  tNumMot (tableau de numéro de mot)
- résultats : temp (variable contenant les items à afficher dans la liste de sélection)
- exceptions :

Règles de traitement

- Précondition :
  la base de données est accessible
  la variable « def » est initialisée.
  le tableau « tNumMot » est initialisé
- Postcondition :
  La liste de sélection IDC_SYN contient tous les synonymes des mots dont le numéro de mot se trouve dans tNumMot
- Excepte :

Requêtes SQL :

- **SQL5**: Cette requête s'exécute pour chaque numéro de mot trouvé par la fonction 3. Elle sélectionne le mot, la catégorie, le genre, la restriction, le commentaire et/ou l'exemple dans les tables correspondant à la langue source, où les numéros de mot et de définition correspondent à ceux de la sélection de l'utilisateur dans la liste de sélection (5).

Cette fonction recherche les synonymes du mot sélectionné par l'utilisateur dans la liste de sélection (5). Elle recherche également la catégorie syntaxique, le genre, les restrictions, les commentaires et les exemples se rapportant à ces synonymes. Ensuite, elle affiche toutes ces

informations dans la liste de sélection (7) : chaque item de la liste de sélection contient un mot suivi de sa catégorie, son genre et parfois des restrictions, des commentaires ou des exemples.

5. void SelectionTraductions (HWND hDlg, LONG int def)

Fonctionnalité : SelectionTraduction

Interface

- arguments : hDlg (identifiant de la boîte de dialogue « spécial dictionnaire multilingue »)
  def (variable contenant un numéro de définition)
  tNumMot (tableau de numéro de mot)
- résultats : temp2 (variable contenant les items à afficher dans la liste de sélection)
- exceptions : message (messages d'erreur informant l'utilisateur qu'on ne trouve pas de traduction pour ce mot dans la langue demandée lorsque c'est le cas.)

Règles de traitement

- Précondition :
  la base de données est accessible
  la variable « def » est initialisée.
  le tableau « tNumMot » est initialisé
- Postcondition :
  La liste de sélection IDC_TRAD contient toutes les traductions des mots dont le numéro de mot se trouve dans tNumMot
- Excepte : message

Requêtes SQL :

- **SQL6**: Cette requête s'exécute pour chaque numéro de mot trouvé par la fonction 3. Elle sélectionne le mot, la catégorie, le genre, la restriction, le commentaire et/ou l'exemple dans les tables correspondant à la langue cible, où les numéros de mot et de définition correspondent à ceux de la sélection de l'utilisateur dans la liste de sélection (5).

Cette fonction recherche les traductions du mot sélectionné par l'utilisateur dans la liste de sélection (5). Elle recherche également la catégorie syntaxique, le genre, les restrictions, les commentaires et les exemples se rapportant à ces traductions. Ensuite, elle affiche toutes ces informations dans la liste de sélection (8) : chaque item de la liste de sélection contient un mot suivi de sa catégorie, son genre et parfois des restrictions, des commentaires ou des exemples.
Cette fonction informe aussi l'utilisateur lorsque des problèmes surgissent.

6. void RechercheBd(HWND hDlg)

Fonctionnalité : RechercheBd

Interface

- arguments : hDlg (identifiant de la boîte de dialogue « spécial dictionnaire multilingue »)
  bufmot (variable contenant le mot à traduire)
- résultats : tNumDef (tableau de numéro de mot)
  liste (mot suivi de sa définition à afficher dans la liste de sélection)
- exceptions :

Règles de traitement

- Précondition :
  la base de données est accessible
  la variable globale « bufmot » est initialisée.
- Postcondition :
  le tableau tNumDef contient tous les numéros de définition répondant à la requête SQL2.
  La liste de sélection IDC_LIST contient les mots, correspondant au numéro de mot de tNumMot, suivi de leur définition correspondant au numéro de définition de tNumDef
  A chaque mot inséré dans la liste IDC_LIST est associé le numéro de définition correspondant.
- Excepte :

Cette fonction appelle les fonctions « SelectionNumeroMotPremier » et « SelectionDefinition ». Elle lance donc la première recherche dans la base de données, une fois que les différents éléments ont été précisés par l'utilisateur.
A la fin de cette fonction, la liste de sélection (5) est complétée à l'écran.

7. void RechercheSelection (HWND hDlg, LONG int numero)

Fonctionnalité : RechercheSelection

Interface

- arguments : hDlg (identifiant de la boîte de dialogue « spécial dictionnaire multilingue »)
  numero (variable contenant un numéro de définition)
- résultats : temp (variable contenant les items à afficher dans la liste de sélection)
  temp2 (variable contenant les items à afficher dans la liste de sélection)
- exceptions :

Règles de traitement

- Précondition :
  la base de données est accessible
  la variable « numero » est initialisée
- Postcondition :
  La liste de sélection IDC_SYN contient tous les synonymes des mots dont le numéro de mot se trouve dans tNumMot
  La liste de sélection IDC_TRAD contient toutes les traductions des mots dont le numéro de mot se trouve dans tNumMot
- Excepte :

Cette fonction est appelée suite à une sélection d'un item dans liste de sélection (5) par l'utilisateur. Elle analyse la sélection et affiche la définition choisie dans le champ prévu à cet effet (6). Ensuite elle appelle les fonctions « SelectionNumeroMot » pour sélectionner le numéro de mot des synonymes, « SelectionSynonymes » pour sélectionner et afficher les synonymes, « SelectionNumeroMot » pour sélectionner le numéro de mot des traductions, « SélectionTraductions » pour sélectionner et afficher les traductions.
A la fin de cette fonction, toutes les informations sont affichées dans la boîte de dialogue « Spécial Dictionnaire multilingue ».

8. BOOL FAR PASCAL _export MyDlgProc (HWND hDlg, UINT message, UINT wParam, LONG lParam)

Fonctionnalité : MyDlgProc

Interface

- arguments : hDlg (identifiant de la boîte de dialogue « spécial dictionnaire multilingue »)
  message, wparam, lparam (arguments qui permettent de saisir ou d'afficher des éléments dans la boîte de dialogue)
- résultats : export (booleen indiquant si la boîte de dialogue est ouverte ou fermée)
- exceptions : message (messages d'erreur apparaissant lorsque les données entrées par l'utilisateur ne sont pas correctes.)

Règles de traitement

- Précondition :
- Postcondition :
  export vaut « false »
- Excepte : message

Cette fonction gère toutes les actions sur la boîte de dialogue « Spécial Dictionnaire multilingue ».

Tout d'abord, elle initialise tous les champs de cette boîte de dialogue.

- Case wm_initdialog:

A l'initialisation de la boîte de dialogue, la fonction remplit les listes de sélection contenant les langues sources et cibles (2)(IDC_LANGUES et IDC_LANGUEC). Ensuite, elle initialise le champ contenant le mot à traduire (1)(IDC_EDIT1): soit elle y inscrit la sélection que l'utilisateur a fait dans le document Interscript, soit elle le remet à blanc.
Si l'utilisateur a déjà fait appel à la boîte de dialogue « Spécial Dictionnaire multilingue » dans sa session Interscript, il a ainsi déjà déterminé la langue source, la langue cible et la langue dans laquelle il désire voir s'afficher les définitions. Au deuxième appel de cette boîte de dialogue, il peut sélectionner un mot à traduire dans le document Interscript. La fonction ayant toutes les informations disponibles à ce moment lancera directement la première recherche dans la base de données, de telle manière que lorsque la boîte de dialogue s'ouvrira, la liste de sélection (5) sera déjà complétée avec les mots contenant le mot à traduire et les définitions.

Ensuite, elle gère toutes les commandes qui surviennent dans cette boîte de dialogue.

- Case wm_command:

Voici les différents événements qui peuvent survenir dans la boîte de dialogue « Spécial Dictionnaire multilingue ».

IDOK : on clique sur le bouton « Information » (4).

Lorsqu'on appuie sur le bouton « information » (4) (IDOK), on déclenche la fonction Recherchebd c'est à dire la première recherche d'informations dans la base de données.
Une série de vérifications sont à faire avant cette recherche.
Il faut d'abord vérifier que les langues source (variable bufsource) et cible (variable bufcible) ainsi que le mot à traduire (variable bufmot) ont été correctement initialisés par l'utilisateur. Par exemple, il ne faut pas que la langue source soit identique à la langue cible. Si c'est le cas, on en informe l'utilisateur.
De plus il faut déterminer en quelle langue on veut avoir les définitions (variable langue). Si l'utilisateur ne l'a pas précisé, on impose une valeur par défaut, soit la langue source.
Avant de lancer la recherche dans la base de données, le curseur se transforme en sablier et montre ainsi que la recherche est en cours. A la fin de la recherche, le curseur redevient une flèche.

- IDCANCEL : on clique sur le bouton « quitter » (9)

Si l'utilisateur a sélectionné un item dans une des listes de sélection contenant les synonymes (7) ou les traductions (8), le mot qu'il a choisi doit apparaître dans le document Interscript une fois la boîte de dialogue « Spécial Dictionnaire multilingue » fermée.
Or en cliquant sur « quitter » (9), on ferme cette boîte de dialogue. Il s'agit donc de préparer le mot à insérer dans le document Interscript et de le stocker dans la variable globale « bufrempl ».
Cette fonction prépare donc le mot à insérer, elle ferme la boîte de dialogue et elle déconnecte ou ferme la base de données.

- IDC_MOT : on entre un mot dans le champ « mot à traduire » (1)

La fonction saisit le mot que l'utilisateur a entré dans le champ correspondant au « mot à traduire » (1) (IDC_MOT). Rappelons que ce mot peut être tapé dans ce champ ou qu'il peut être sélectionné dans le document Interscript ce qui entraîne son affichage dans le champ « mot à traduire » (1).
La fonction stocke le mot saisi dans une variable « bufmot ».

- IDC_LANGUES : on choisit une langue source

La fonction saisit la langue source sélectionnée par l'utilisateur dans la liste de sélection correspondante (2) (IDC_LANGUES). Cette fonction stocke ensuite la langue source choisie dans une variable « bufsource ».
De plus, la fonction initialise l'en-tête de la liste de sélection (IDC_TRADS) contenant les synonymes (7) en y inscrivant la langue source contenue dans « bufsource ».

- IDC_LANGUEC : on choisit une langue cible

La fonction saisit la langue cible sélectionnée par l'utilisateur dans la liste de sélection correspondante (2) (IDC_LANGUEC). Cette fonction stocke ensuite la langue cible choisie dans une variable « bufcible ».
De plus, la fonction initialise l'en-tête de la liste de sélection (IDC_TRADC) contenant les traductions (8) en y inscrivant la langue cible contenue dans « bufcible ».

- IDC_RADIO1 : on sélectionne 'langue source' pour l'affichage des définitions

A partir du moment ou l'utilisateur a cliqué sur un des boutons radio (3), on sait en quelle langue les définitions vont apparaître.
Donc, si le bouton radio langue source (IDC_RADIO1) est sélectionné, la fonction stocke la variable « bufsource » dans une variable « langue » qui indiquera que la fonction « SelectionDefinition » doit afficher les définitions en langue source.

- IDC_RADIO2 : on sélectionne 'langue cible' pour l'affichage des définitions

Si le bouton radio langue cible (IDC_RADIO2) est sélectionné (3), la fonction stocke la variable « bufcible » dans une variable « langue » qui indiquera que la fonction « SelectionDefinition » doit afficher les définitions en langue cible.

- IDC_LIST : on clique sur un item de la liste de sélection (5)

Lorsque l'utilisateur sélectionne d'un item dans la liste de sélection (5) (IDC_LIST), la fonction retrouve le numéro de définition associé à cette sélection et le stocke dans la variable « numero ». Ensuite, la fonction appelle la fonction « RechercheSelection » avec ce numéro qui aidera la recherche des synonymes et traductions du mot sélectionné.

- IDC_SYN : on clique sur un item de la liste de sélection (7)

Si l'utilisateur sélectionne un item dans la liste de sélection (7) (IDC_SYN), c'est qu'il désire insérer un synonyme dans le document Interscript. La fonction va donc stocker l'item sélectionné dans une variable « bufrempl ». Le synonyme sélectionné sera inséré dans le document Interscript à la fermeture de la boîte de dialogue.

- IDC_TRAD : on clique sur un item de la liste de sélection (8)

Si l'utilisateur sélectionne un item dans la liste de sélection (8) (IDC_TRAD), c'est qu'il désire insérer une traduction dans le document Interscript. La fonction va donc stocker l'item sélectionné dans une variable « bufrempl ». La traduction sélectionnée sera insérée dans le document Interscript à la fermeture de la boîte de dialogue.

Voici maintenant les fonctions qui ont été fournies par Interscript

9. GLOBAL void PFAR TpsCallService(LONG lVal, WORD wService)

Cette fonction demande à une DLL de lui fournir ses services. Toute la DLL est donc à la disposition d'Interscript.

Cette fonction commence par sauvegarder la position du curseur dans le document Interscript. En effet, elle doit savoir où elle va éventuellement insérer un synonyme ou une traduction sélectionné par l'utilisateur dans la liste de sélection (7) ou (8). Car c'est cette fonction qui insère la sélection dans le document Interscript.
Si l'utilisateur a sélectionné un mot dans le document Interscript, ce mot est stocké dans une variable globale « azBuffer » accessible à la DLL, afin de pouvoir l'afficher dans la boîte de dialogue « Spécial Dictionnaire multilingue » dans le champ correspondant à « mot à traduire » (1).
C'est dans cette fonction que se fait la connexion avec la base de données. Si cette connexion ne se fait pas correctement, la fonction affiche un message d'erreur pour prévenir l'utilisateur qu'il a mal configuré ODBC.
Une fois la connexion de la base de données établies, la fonction ouvre la boîte de dialogue « Spécial Dictionnaire multilingue ». La fonction fait donc appel à la DLL du dictionnaire multilingue.
Une fois que l'utilisateur a cliqué sur le bouton « quitter » (9) de la boîte de dialogue, la procédure « TpsCallSevice » reprend la main.
Si l'utilisateur a sélectionné un item dans une des listes de sélection contenant les synonymes ou les traductions (7) ou (8), la fonction insère cette sélection, qui se trouve dans la variable

globale « bufrempl », dans le document Interscript à la position du curseur qu'elle a sauvegardé.
La dernière chose à faire est de remettre les tampons à blanc pour que le champ correspondant à mot à traduire (1) soit vide ou prêt à recevoir de l'information à la réouverture la boîte de dialogue « Spécial Dictionnaire multilingue ».

10. GLOBAL void PFAR TpsNotifyEvent (LONG lVal, WORD wEvent, LONG lParam1, LONG lParam2)

Cette fonction informe la DLL qu'un événement vient de se produire

Cette fonction sert à insérer l'item de menu « Dictionnaire Multilingue » dans le menu « Spécial » d'Interscript.

11. GLOBAL void PFAR TpsProc (WORD wSrvFunctionId, TS_MISC pSrvFunction)

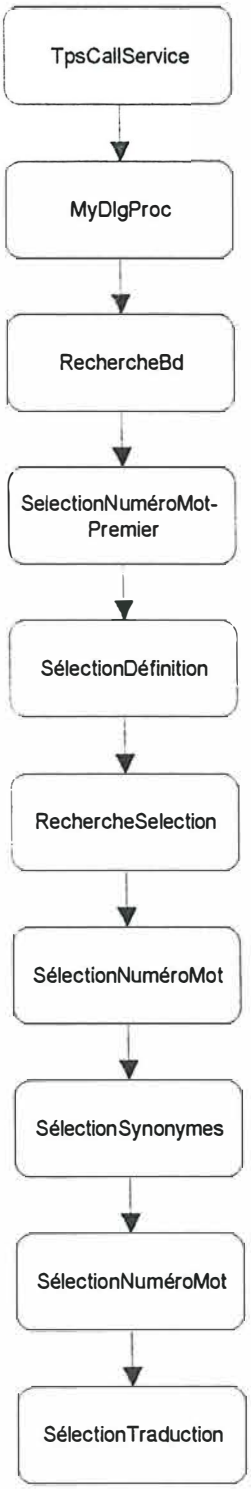
Cette fonction enregistre les fonctions qu'Interscript à fournit à la DLL.

Cette fonction met à la disposition de la DLL toute un série de fonctions Interscript.

12. GLOBAL int PFAR TpsQueryServiceCount (void)

Cette fonction détermine le nombre de service que la DLL externe offre.

Toutes les fonctions ayant été décrites, voici comment elles s'enchaînent lors de l'exécution de notre exemple.



Conclusion

Ce mémoire a voulu expliciter le développement d'un dictionnaire multilingue de consultation conçu pour le projet Apollo. Il décrit chaque partie de ce dictionnaire ainsi que son champs d'application.

La description commence par le traitement de l'aspect linguistique de cet outil. A l'aide de la classification du premier chapitre, on a pu déterminer que le dictionnaire multilingue du projet Apollo est un dictionnaire utilisé par les personnes et que ce dictionnaire est un outil intégré dans le traitement de textes Interscript.
On a aussi été convaincu, après l'analyse détaillée des fichiers Spectrum et ABB qui sont les ressources principales du lexique utilisé par le dictionnaire multilingue, que les données que l'on trouve dans le dictionnaire Apollo sont celles qu'on pourrait s'attendre à retrouver dans ce type de dictionnaire.
Les fichiers textes Spectrum et ABB ont une structure spéciale qui convient aussi bien au traducteur automatique qu'au dictionnaire multilingue. On a donc observé que certains champs des fichiers textes ne sont pas considérés du côté du dictionnaire et sont utiles uniquement pour le traducteur automatique.

Le deuxième chapitre a décrit différentes ressources textuelles qui peuvent faire l'objet d'un corpus. A nouveau, on a pu classifier les ressources utilisées pour le projet Apollo. On a rassemblé des ressources textuelles d'une part telles que des cours boursiers et des ressources lexicographiques telles que les fichiers Spectrum et ABB d'autre part. Ces ressources, structurées ou non peuvent être considérée comme scolaires si on suit la classification de Gunnel Engwall [ATKINS] ou comme faisant partie d'un sous-langage définit par Guy Deville dans [DEVILLE].

La troisième partie du mémoire parle du projet Apollo lui-même. On y a décrit les différents partenaires qui ont participé au projet et on y a détaillé le rôle de chacun.
Ce chapitre a aussi expliqué l'interaction entre les différents éléments du projet. On a vu que l'utilisateur n'interagit qu'avec le traitement de textes Interscript et qu'il ne peut donc accéder au module de traduction automatique, au module de gestion de versions et au dictionnaire multilingue qu'à partir du traitement de textes.
On a observé également les modifications qu'on a dû apporter au traitement de textes pour qu'il puisse traiter des documents multilingues.

C'est également dans ce chapitre qu'on a étudié le processus de traduction de CAT2.
Lors de l'étude de l'architecture technique du projet, il a été précisé que ce projet s'est développé sous P.C. et sous UNIX.

Le dictionnaire multilingue, décrit dans le dernier chapitre, quant à lui a été développé sous P.C. On a vu que certains choix ont été faits quant à la structure des bases de données pour rendre celles-ci les plus générales possible par rapport à d'autres dictionnaires. Cependant nous ne possédions pas toute l'information nécessaire pour remplir chaque champs de la base de données.
Pour l'interface, il a été décidé qu'une seule boîte de dialogue devait donner le plus d'informations possibles pour rendre les choses claires en un seul coup d'oeil.

Cet outil Apollo me parait très utile pour des utilisateurs qui ne sont pas des traducteurs professionnels. Il est intéressant de pouvoir traduire un texte à partir de sa version électronique à l'aide d'un traitement de textes convivial. Mais il est également important de pouvoir comparer les textes sources et cibles entre eux.

Quant au dictionnaire multilingue, il constitue une aide précieuse à l'utilisateur qui n'est pas certain de la traduction que lui a fournie le traducteur automatique. Cet outil servira aussi à un utilisateur qui lit un texte dans une langue qu'il ne maîtrise pas bien et qui pourra alors demander au dictionnaire certaines explications à propos de différents termes financiers ou bancaires.

[DEVILLE] « The complexity of natural language has become very obvious since computer specialists started to use their instruments for non-mathematical applications. Up till now, no computer has been able to treat language in its entirety. »

Références bibliographiques

[ATKINS] B. T. S. Atkins , A. Zampolli, *Computational Approaches to the lexicon*, Clarendon Press Oxford, 1994.

[BERLEUR] J. Berleur s.j., *Ordinateurs et Langages : des « Novlangues » à l'Horizon ?*, Enjeux N°. 12, mai 1987.

[BOUILLON] P. Bouillon, A. Clas, *La Traductique*, Press de l'université de Montréal, 1993.

[CEUSTERS] W. Ceusters, G. Deville, E. Herbigniaux, P. Mousel, O. Streiter, G. Thienpont, *IAI Working Paper N°31 : The Anthem Prototype*, Saarbrücken, 1994.

[DEVILLE] G. Deville, *Modelization of Task-oriented Utterances in a Man-machine Dialogue System Part 1*, Thèse de Doctorat, 1989.

[DEVILLE_1] G. Deville, E. Herbigniaux, O. Steiter, *Anthem Advanced Natural Language Interface for Multilingual Text Generation in Healthcare The CAT2 Interface Stucture Applied to the Medical Diagnose Sublanguage*, Namur, 1994

[DEVILLE_2] G. Deville, E. Herbigniaux, *APOLLO Project Review Text of the Slides presented by the FUNDP (Linguistic Issues)*, APOLLO Deliverable LE-1033L-D-01.

[DEVILLE_3] G. Deville, E. Herbigniaux, *Preparatory Document for the APOLLO TM1 Technical Meeting to be held on January 18th, 1996 at the CRP-CU, Luxembourg*, APOLLO report 001 version 1.0

[DEVILLE_4] G. Deville, E. Herbigniaux, V. Thunus, *Elaboration of an on line Multilingual Dictionary for the APOLLO Workbench Mock Up*, APOLLO report 006 version 2.0

[HEYMANS] P. Heymans, M. Rouard, *La Traduction Automatique Etude de cas : Logos*, Mémoire, 1994.

[HUTCHINS] W. J. Hutchins, H.L. Somers, *An Introduction to Machine Translation*, Academic Press, 1992.

[LAROUSSE] *Le Petit Larousse Illustré*, Larousse, 1993.

[MOUSEL_1] P. Mousel, *Functional Specifications of the APOLLO Prototype*, APOLLO deliverable LE-1033L-D-03.

[MOUSEL_2] P. Mousel, *Technical Specifications of the APOLLO Prototype*, APOLLO deliverable LE-1033L-D-04.

[SHARP] R. Sharp, *IAI Working Paper N°27 : CAT2 Reference Manual Version 3.6*, Saarbrücken, 1994.

[SABAH] G. Sabah, *L'Intelligence Artificielle et le Language : Représentation des Connaissances*, Hermès, 1988.

[WALKER] D. E. Walker, A. Zampolli, N.Calzolari, *Automating the Lexicon Research and Practice in a Multilingual Environment*, Clarendon Press Oxford, 1995.

Annexes

Annexe 1 : Le code C qui a servi à remplir la base de données

```
/*
 * file InitBdSp.c
 */

#include <global.h>
#include <windows.h>
#include "sql.h"
#include "sqlext.h"
#include <string.h>
#include <stdio.h>
#include <malloc.h>
#define         MAX_BUF_SIZE 256


MLOCAL BYTE * PNEAR BuildString(FILE * fichier);


void InitialiseBd ()
{
HENV henv;
HDBC hdbc;
HSTMT hstmt;
RETCODE rc;
FILE *fichier;
BYTE car [2];
LOCAL BYTE * def=NULL ;
LOCAL BYTE mot [255];
LOCAL BYTE gen [32];
LOCAL BYTE cat [32];
LOCAL BYTE ae_be [32];
BYTE *pc;
LOCAL BYTE com [255];
LOCAL BYTE res [255];
LOCAL BYTE abrev [50];
SDWORD cbvalue = SQL_NTS;
SDWORD cbnbr = 0;
LONG int compteur_def = 0;
LONG int compteur_an = 0;
LONG int compteur_fr = 0;
LONG int compteur_al = 0;
LONG int compteur_n = 0;
```

```
LONG int numero ;
SDWORD cbnumero;
int i,j,k;

fichier = fopen("test.txt","r");


car[0] =' ';
car[ 1] ='\0';


if (fichier==NULL)
        MessageBox(NULL,"pas d'ouverture de fichier",NULL,MB_OK);


rc = SQLAllocEnv(&henv);
rc = SQLAllocConnect(henv,&hdbc);
rc = SQLConnect(hdbc,"Apollo4",SQL_NTS,"",SQL_NTS,"",SQL_NTS);

car[0] = getc(fichier);

while (!feof(fichier))

{
        rc = SQLAllocStmt(hdbc,&hstmt);

        /*initialisation du no def*/
        compteur_def=compteur_def+1;
        /*lecture du champ 001*/
        while (car[0] != '\n')
        {
                car[0] = getc(fichier);
        }
        /*lecture du champ 002*/
        car[0] = getc(fichier);
        while (car[0] != '\n')
        {
                car[0] = getc(fichier);
        }
        /*lecture de la définition*/
        def = BuildString(fichier);

        rc = SQLPrepare(hstmt,"INSERT INTO DEFINITION (NUM_DEF,DEF_N)
VALUES (?,?)",SQL_NTS);
        rc =
SQLBindParameter(hstmt,1,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_DOUBLE,0,0,&c
ompteur_def,0,&cbvalue);
        rc =
SQLBindParameter(hstmt,2,SQL_PARAM_INPUT,SQL_C_CHAR,SQL_LONGVARCHAR,
1050,1050,def,1050,&cbvalue);
```

```
        rc = SQLExecute(hstmt);
        rc = SQLFreeStmt(hstmt, SQL_DROP);

        free (def);
        def=NULL;


        /*lecture des champs 004 a 020*/
        for (i=1;i<18;i++)
        {
                car[0] = getc(fichier);
                while (car[0] != '\n')
                {
                        car[0] = getc(fichier);
                }
        }

/*****anglais*****/
        for (j=1;j<5;j++)
        {

                /*lecture mot angl*/
                car[0] = getc(fichier);
                strcpy(mot,&car[0]);
                while (car[0] != '\n')
                {
                        car[0] = getc(fichier);
                        strcat(mot,&car[0]);
                }
                pc = strchr(mot,'\n');
                *pc=0;

                if (mot[0]!='-')
                {
                        /*lecture pluriel*/
                        car[0] = getc(fichier);
                        while (car[0] != '\n')
                        {
                                car[0] = getc(fichier);
                        }
                        /*lecture de la catégorie*/
                        car[0] = getc(fichier);
                        strcpy(cat,&car[0]);
                        while (car[0] != '\n')
                        {
                                car[0] = getc(fichier);
                                strcat(cat,&car[0]);
                        }
                        pc = strchr(cat,'\n');
                        *pc=0;
                        /*lecture de ae-be*/
```

```
car[0] = getc(fichier);
strcpy(ae_be,&car[0]);
while (car[0] != '+')
{
        car[0] = getc(fichier);
        strcat(ae_be,&car[0]);
}
pc = strchr(ae_be,'+');
pc = pc-1;
*pc=0;
/*lecture des comment*/
car[0] = getc(fichier);
car[0] = getc(fichier);
strcpy(com,&car[0]);
while (car[0] != '+')
{
        car[0] = getc(fichier);
        strcat(com,&car[0]);
}
pc = strchr(com,'+');
pc=pc-1;
*pc=0;
/*lecture des restrictions*/
car[0] = getc(fichier);
car[0] = getc(fichier);
strcpy(res,&car[0]);
while (car[0] != '\n')
{
        car[0] = getc(fichier);
        strcat(res,&car[0]);
}
pc = strchr(res,'\n');
*pc=0;
/*lecture du 5ieme champ du groupe (inutile)*/
car[0] = getc(fichier);
while (car[0] != '\n')
{
        car[0] = getc(fichier);
}

rc = SQLAllocStmt(hdbc,&hstmt);

rc = SQLPrepare(hstmt,"SELECT num_mot FROM Anglais WHERE
(mot = ?) AND (categorie = ?) AND (ae_be = ?)",SQL_NTS);
rc =
SQLBindParameter(hstmt,1,SQL_PARAM_INPUT_OUTPUT,SQL_C_CHAR,SQL_VARCH
AR,255,255,mot,255,&cbvalue);
rc =
SQLBindParameter(hstmt,2,SQL_PARAM_INPUT_OUTPUT,SQL_C_CHAR,SQL_VARCH
AR,32,32,cat,32,&cbvalue);
```

```
                    rc =
SQLBindParameter(hstmt,3,SQL_PARAM_INPUT_OUTPUT,SQL_C_CHAR,SQL_VARCH
AR,32,32,ae_be,32,&cbvalue);
                    rc = SQLExecute(hstmt);

                    rc = SQLBindCol(hstmt,1,SQL_C_SLONG,&numero,0,&cbnumero);
                    rc = SQLFetch(hstmt);

                    SQLFreeStmt(hstmt, SQL_DROP);


                    SQLAllocStmt(hdbc,&hstmt);


                    if (rc==SQL_NO_DATA_FOUND)
                    {
                            compteur_an=compteur_an+1;
                            rc = SQLPrepare(hstmt,"INSERT INTO ANGLAIS VALUES
(?,?,?,?)",SQL_NTS);
                            rc =
SQLBindParameter(hstmt,1,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_DOUBLE,0,0,&c
ompteur_an,0,&cbvalue);
                            rc =
SQLBindParameter(hstmt,2,SQL_PARAM_INPUT,SQL_C_DEFAULT,SQL_LONGVARC
HAR,255,255,mot,255,&cbvalue);
                            rc =
SQLBindParameter(hstmt,3,SQL_PARAM_INPUT,SQL_C_DEFAULT,SQL_VARCHAR,32
,32,cat,32,&cbvalue);
                            rc =
SQLBindParameter(hstmt,4,SQL_PARAM_INPUT,SQL_C_DEFAULT,SQL_VARCHAR,32
,32,ae_be,32,&cbvalue);

                            numero=compteur_an;

                            rc = SQLExecute(hstmt);
                    }
                    rc = SQLFreeStmt(hstmt, SQL_DROP);

                    rc = SQLAllocStmt(hdbc,&hstmt);

                    rc = SQLPrepare(hstmt,"INSERT INTO TRADUCTION_AN
(NUM_DEF,NUM_MOT,COMMENT,RESTRICTION) VALUES (?,?,?,?)",SQL_NTS);
                    rc =
SQLBindParameter(hstmt,1,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_DOUBLE,0,0,&c
ompteur_def,0,&cbvalue);
                    rc =
SQLBindParameter(hstmt,2,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_DOUBLE,0,0,&n
umero,0,&cbvalue);
                    rc =
SQLBindParameter(hstmt,3,SQL_PARAM_INPUT,SQL_C_DEFAULT,SQL_LONGVARC
HAR,255,255,com,255,&cbvalue);
```

```
                    rc =
SQLBindParameter(hstmt,4,SQL_PARAM_INPUT,SQL_C_DEFAULT,SQL_LONGVARC
HAR,255,255,res,255,&cbvalue);

                    rc = SQLExecute(hstmt);


                    rc = SQLFreeStmt(hstmt, SQL_DROP);
            }
            else
            {
                    car[0] = getc(fichier);
                    while (car[0] != '4')
                    {
                            car[0] = getc(fichier);
                    }
                    car[0] = getc(fichier);
                    car[0] = getc(fichier);
                    j=5;
            }
    }



/*****francais*****/
    for (j=1;j<5;j++)
    {

            /*lecture mot */
            car[0] = getc(fichier);
            strcpy(mot,&car[0]);
            while (car[0] != '\n')
            {
                    car[0] = getc(fichier);
                    strcat(mot,&car[0]);
            }
            pc = strchr(mot,'\n');
            *pc=0;
            if (mot[0]!='-')
            {
                    /*lecture genre*/
                    car[0] = getc(fichier);
                    strcpy(gen,&car[0]);
                    while (car[0] != '\n')
                    {
                            car[0] = getc(fichier);
                            strcat(gen,&car[0]);
                    }
                    pc = strchr(gen,'\n');
                    *pc=0;
```

```
/*lecture de la catégorie*/
car[0] = getc(fichier);
strcpy(cat,&car[0]);
while (car[0] != '\n')
{
        car[0] = getc(fichier);
        strcat(cat,&car[0]);
}
pc = strchr(cat,'\n');
*pc=0;
/*lecture des comment*/
car[0] = getc(fichier);
strcpy(com,&car[0]);
while (car[0] != '+')
{
        car[0] = getc(fichier);
        strcat(com,&car[0]);
}
pc = strchr(com,'+');
pc=pc-1;
*pc=0;
/*lecture des restrictions*/
car[0] = getc(fichier);
car[0] = getc(fichier);
strcpy(res,&car[0]);
while (car[0] != '\n')
{
        car[0] = getc(fichier);
        strcat(res,&car[0]);
}
pc = strchr(res,'\n');
*pc=0;
/*lecture du 5ieme champ du groupe (inutile)*/
car[0] = getc(fichier);
while (car[0] != '\n')
{
        car[0] = getc(fichier);
}

rc = SQLAllocStmt(hdbc,&hstmt);

rc = SQLPrepare(hstmt,"SELECT num_mot FROM Français WHERE
(mot = ?) AND (categorie = ?) AND (genre = ?)",SQL_NTS);
rc =
SQLBindParameter(hstmt,1,SQL_PARAM_INPUT_OUTPUT,SQL_C_CHAR,SQL_VARCH
AR,255,255,mot,255,&cbvalue);
rc =
SQLBindParameter(hstmt,2,SQL_PARAM_INPUT_OUTPUT,SQL_C_CHAR,SQL_VARCH
AR,32,32,cat,32,&cbvalue);
```

```
                    rc =
SQLBindParameter(hstmt,3,SQL_PARAM_INPUT_OUTPUT,SQL_C_CHAR,SQL_VARCH
AR,32,32,gen,32,&cbvalue);
                    rc = SQLExecute(hstmt);

                    rc = SQLBindCol(hstmt,1,SQL_C_SLONG,&numero,0,&cbnumero);
                    rc = SQLFetch(hstmt);

                    SQLFreeStmt(hstmt, SQL_DROP);


                    SQLAllocStmt(hdbc,&hstmt);


                    if (rc==SQL_NO_DATA_FOUND)
                    {
                            compteur_fr=compteur_fr+1;
                            rc = SQLPrepare(hstmt,"INSERT INTO Français VALUES
(?,?,?,?)",SQL_NTS);
                            rc =
SQLBindParameter(hstmt,1,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_DOUBLE,0,0,&c
ompteur_fr,0,&cbvalue);
                            rc =
SQLBindParameter(hstmt,2,SQL_PARAM_INPUT,SQL_C_DEFAULT,SQL_LONGVARC
HAR,255,255,mot,255,&cbvalue);
                            rc =
SQLBindParameter(hstmt,3,SQL_PARAM_INPUT,SQL_C_DEFAULT,SQL_VARCHAR,32
,32,cat,32,&cbvalue);
                            rc =
SQLBindParameter(hstmt,4,SQL_PARAM_INPUT,SQL_C_DEFAULT,SQL_VARCHAR,32
,32,gen,32,&cbvalue);

                            numero=compteur_fr;

                            rc = SQLExecute(hstmt);

                    }
                    rc = SQLFreeStmt(hstmt, SQL_DROP);

                    rc = SQLAllocStmt(hdbc,&hstmt);

                    rc = SQLPrepare(hstmt,"INSERT INTO TRADUCTION_F
(NUM_DEF,NUM_MOT,COMMENT,RESTRICTION) VALUES (?,?,?,?)",SQL_NTS);
                    rc =
SQLBindParameter(hstmt,1,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_DOUBLE,0,0,&c
ompteur_def,0,&cbvalue);
                    rc =
SQLBindParameter(hstmt,2,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_DOUBLE,0,0,&n
umero,0,&cbvalue);
```

```
                    rc =
SQLBindParameter(hstmt,3,SQL_PARAM_INPUT,SQL_C_DEFAULT,SQL_LONGVARC
HAR,255,255,com,255,&cbvalue);
                    rc =
SQLBindParameter(hstmt,4,SQL_PARAM_INPUT,SQL_C_DEFAULT,SQL_LONGVARC
HAR,255,255,res,255,&cbvalue);

                    rc = SQLExecute(hstmt);
                    rc = SQLFreeStmt(hstmt, SQL_DROP);
            }
            else
            {
                    car[0] = getc(fichier);
                    while (car[0] != '6')
                    {
                            car[0] = getc(fichier);
                    }
                    car[0] = getc(fichier);
                    car[0] = getc(fichier);
                    j=5;
            }

    }

/*****Néerlandais*****/
    for (j=1;j<5;j++)
    {

            /*lecture mot */
            car[0] = getc(fichier);
            strcpy(mot,&car[0]);
            while (car[0] != '\n')
            {
                    car[0] = getc(fichier);
                    strcat(mot,&car[0]);
            }
            pc = strchr(mot,'\n');
            *pc=0;
            if (mot[0]!='-')
            {
                    /*lecture genre*/
                    car[0] = getc(fichier);
                    strcpy(gen,&car[0]);
                    while (car[0] != '\n')
                    {
                            car[0] = getc(fichier);
                            strcat(gen,&car[0]);
                    }
                    pc = strchr(gen,'\n');
                    *pc=0;
                    /*lecture de la catégorie*/
```

```
car[0] = getc(fichier);
strcpy(cat,&car[0]);
while (car[0] != '\n')
{
        car[0] = getc(fichier);
        strcat(cat,&car[0]);
}
pc = strchr(cat,'\n');
*pc=0;
/*lecture du 4ieme champ du groupe (inutile)*/
car[0] = getc(fichier);
while (car[0] != '\n')
{
        car[0] = getc(fichier);
}
/*abréviation*/
car[0] = getc(fichier);
if (car[0] != '0')
{
        strcpy(abrev,&car[0]);
        while (car[0] != '\n')
        {
                car[0] = getc(fichier);
                strcat(abrev,&car[0]);
        }
        pc = strchr(abrev,'\n');
        *pc=0;
}
else
{
        while (car[0] != '\n')
        {
                car[0] = getc(fichier);
        }
}

rc = SQLAllocStmt(hdbc,&hstmt);
rc = SQLPrepare(hstmt,"SELECT num_mot FROM Néerld WHERE
(mot = ?) AND (categorie = ?) AND (genre = ?)",SQL_NTS);
rc =
SQLBindParameter(hstmt,1,SQL_PARAM_INPUT_OUTPUT,SQL_C_CHAR,SQL_VARCH
AR,255,255,mot,255,&cbvalue);
rc =
SQLBindParameter(hstmt,2,SQL_PARAM_INPUT_OUTPUT,SQL_C_CHAR,SQL_VARCH
AR,32,32,cat,32,&cbvalue);
rc =
SQLBindParameter(hstmt,3,SQL_PARAM_INPUT_OUTPUT,SQL_C_CHAR,SQL_VARCH
AR,32,32,gen,32,&cbvalue);
rc = SQLExecute(hstmt);

rc = SQLBindCol(hstmt,1,SQL_C_SLONG,&numero,0,&cbnumero);
```

```
                    rc = SQLFetch(hstmt);

                    SQLFreeStmt(hstmt, SQL_DROP);


                    SQLAllocStmt(hdbc,&hstmt);


                    if (rc==SQL_NO_DATA_FOUND)
                    {
                            compteur_n=compteur_n+1;
                            rc = SQLPrepare(hstmt,"INSERT INTO Néerld VALUES
(?,?,?,?)",SQL_NTS);
                            rc =
SQLBindParameter(hstmt,1,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_DOUBLE,0,0,&c
ompteur_n,0,&cbvalue);
                            rc =
SQLBindParameter(hstmt,2,SQL_PARAM_INPUT,SQL_C_DEFAULT,SQL_LONGVARC
HAR,255,255,mot,255,&cbvalue);
                            rc =
SQLBindParameter(hstmt,3,SQL_PARAM_INPUT,SQL_C_DEFAULT,SQL_VARCHAR,32
,32,cat,32,&cbvalue);
                            rc =
SQLBindParameter(hstmt,4,SQL_PARAM_INPUT,SQL_C_DEFAULT,SQL_VARCHAR,32
,32,gen,32,&cbvalue);

                            numero=compteur_n;

                            rc = SQLExecute(hstmt);
                    }
                    rc = SQLFreeStmt(hstmt, SQL_DROP);

                    rc = SQLAllocStmt(hdbc,&hstmt);

                    rc = SQLPrepare(hstmt,"INSERT INTO TRADUCTION_N
(NUM_DEF,NUM_MOT) VALUES (?,?)",SQL_NTS);
                    rc =
SQLBindParameter(hstmt,1,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_DOUBLE,0,0,&c
ompteur_def,0,&cbvalue);
                    rc =
SQLBindParameter(hstmt,2,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_DOUBLE,0,0,&n
umero,0,&cbvalue);

                    rc = SQLExecute(hstmt);

                    rc = SQLFreeStmt(hstmt, SQL_DROP);


                    if (strcmp(abrev,"afkorting")!=0)
                    {
                            SQLAllocStmt(hdbc,&hstmt);
```

```
                        compteur_n=compteur_n+1;
                        rc = SQLPrepare(hstmt,"INSERT INTO Néerld VALUES
(?,?,?,?)",SQL_NTS);
                        rc =
SQLBindParameter(hstmt,1,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_DOUBLE,0,0,&c
ompteur_n,0,&cbvalue);
                        rc =
SQLBindParameter(hstmt,2,SQL_PARAM_INPUT,SQL_C_DEFAULT,SQL_LONGVARC
HAR,255,255,abrev,255,&cbvalue);
                        rc =
SQLBindParameter(hstmt,3,SQL_PARAM_INPUT,SQL_C_DEFAULT,SQL_VARCHAR,32
,32,cat,32,&cbvalue);
                        rc =
SQLBindParameter(hstmt,4,SQL_PARAM_INPUT,SQL_C_DEFAULT,SQL_VARCHAR,32
,32,gen,32,&cbvalue);

                        numero=compteur_n;

                        rc = SQLExecute(hstmt);

                        rc = SQLFreeStmt(hstmt, SQL_DROP);

                        rc = SQLAllocStmt(hdbc,&hstmt);

                        rc = SQLPrepare(hstmt,"INSERT INTO TRADUCTION_N
(NUM_DEF,NUM_MOT) VALUES (?,?)",SQL_NTS);
                        rc =
SQLBindParameter(hstmt,1,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_DOUBLE,0,0,&c
ompteur_def,0,&cbvalue);
                        rc =
SQLBindParameter(hstmt,2,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_DOUBLE,0,0,&n
umero,0,&cbvalue);

                        rc = SQLExecute(hstmt);

                        rc = SQLFreeStmt(hstmt, SQL_DROP);
                }
        }
        else
        {
                car[0] = getc(fichier);
                while (car[0] != '8')
                {
                        car[0] = getc(fichier);
                }
                car[0] = getc(fichier);
                car[0] = getc(fichier);
                j=5;
        }
```

```
        }


/*****allemand*****/
        for (j=1;j<5;j++)
        {

                /*lecture mot */
                car[0] = getc(fichier);
                strcpy(mot,&car[0]);
                while (car[0] != '\n')
                {
                        car[0] = getc(fichier);
                        strcat(mot,&car[0]);
                }
                pc = strchr(mot,'\n');
                *pc=0;
                if (mot[0]!='-')
                {
                        /*lecture genre*/
                        car[0] = getc(fichier);
                        strcpy(gen,&car[0]);
                        while (car[0] != '\n')
                        {
                                car[0] = getc(fichier);
                                strcat(gen,&car[0]);
                        }
                        pc = strchr(gen,'\n');
                        *pc=0;
                        /*lecture de la catégorie*/
                        car[0] = getc(fichier);
                        strcpy(cat,&car[0]);
                        while (car[0] != '\n')
                        {
                                car[0] = getc(fichier);
                                strcat(cat,&car[0]);
                        }
                        pc = strchr(cat,'\n');
                        *pc=0;
                        /*lecture des comment*/
                        car[0] = getc(fichier);
                        strcpy(com,&car[0]);
                        while (car[0] != '+')
                        {
                                car[0] = getc(fichier);
                                strcat(com,&car[0]);
                        }
                        pc = strchr(com,'+');
                        pc=pc-1;
                        *pc=0;
```

```
                    /*lecture des restrictions*/
                    car[0] = getc(fichier);
                    car[0] = getc(fichier);
                    strcpy(res,&car[0]);
                    while (car[0] != '\n')
                    {
                            car[0] = getc(fichier);
                            strcat(res,&car[0]);
                    }
                    pc = strchr(res,'\n');
                    *pc=0;
                    /*lecture du 5ieme champ du groupe (inutile)*/
                    car[0] = getc(fichier);
                    while (car[0] != '\n')
                    {
                            car[0] = getc(fichier);
                    }

                    rc = SQLAllocStmt(hdbc,&hstmt);

                    rc = SQLPrepare(hstmt,"SELECT num_mot FROM Allemand WHERE
(mot = ?) AND (categorie = ?) AND (genre = ?)",SQL_NTS);
                    rc =
SQLBindParameter(hstmt,1,SQL_PARAM_INPUT_OUTPUT,SQL_C_CHAR,SQL_VARCH
AR,255,255,mot,255,&cbvalue);
                    rc =
SQLBindParameter(hstmt,2,SQL_PARAM_INPUT_OUTPUT,SQL_C_CHAR,SQL_VARCH
AR,32,32,cat,32,&cbvalue);
                    rc =
SQLBindParameter(hstmt,3,SQL_PARAM_INPUT_OUTPUT,SQL_C_CHAR,SQL_VARCH
AR,32,32,gen,32,&cbvalue);
                    rc = SQLExecute(hstmt);

                    rc = SQLBindCol(hstmt,1,SQL_C_SLONG,&numero,0,&cbnumero);
                    rc = SQLFetch(hstmt);

                    SQLFreeStmt(hstmt, SQL_DROP);


                    SQLAllocStmt(hdbc,&hstmt);


                    if (rc==SQL_NO_DATA_FOUND)
                    {
                            compteur_al=compteur_al+1;
                            rc = SQLPrepare(hstmt,"INSERT INTO Allemand VALUES
(?,?,?,?)",SQL_NTS);
                            rc =
SQLBindParameter(hstmt,1,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_DOUBLE,0,0,&c
ompteur_al,0,&cbvalue);
```

```
                        rc =
SQLBindParameter(hstmt,2,SQL_PARAM_INPUT,SQL_C_DEFAULT,SQL_LONGVARC
HAR,255,255,mot,255,&cbvalue);
                        rc =
SQLBindParameter(hstmt,3,SQL_PARAM_INPUT,SQL_C_DEFAULT,SQL_VARCHAR,32
,32,cat,32,&cbvalue);
                        rc =
SQLBindParameter(hstmt,4,SQL_PARAM_INPUT,SQL_C_DEFAULT,SQL_VARCHAR,32
,32,gen,32,&cbvalue);

                        numero=compteur_al;

                        rc = SQLExecute(hstmt);
                }

                rc = SQLFreeStmt(hstmt, SQL_DROP);

                rc = SQLAllocStmt(hdbc,&hstmt);

                rc = SQLPrepare(hstmt,"INSERT INTO TRADUCTION_AL
(NUM_DEF,NUM_MOT,COMMENT,RESTRICTION) VALUES (?,?,?,?)",SQL_NTS);
                rc =
SQLBindParameter(hstmt,1,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_DOUBLE,0,0,&c
ompteur_def,0,&cbvalue);
                rc =
SQLBindParameter(hstmt,2,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_DOUBLE,0,0,&n
umero,0,&cbvalue);
                rc =
SQLBindParameter(hstmt,3,SQL_PARAM_INPUT,SQL_C_DEFAULT,SQL_LONGVARC
HAR,255,255,com,255,&cbvalue);
                rc =
SQLBindParameter(hstmt,4,SQL_PARAM_INPUT,SQL_C_DEFAULT,SQL_LONGVARC
HAR,255,255,res,255,&cbvalue);

                rc = SQLExecute(hstmt);


                rc = SQLFreeStmt(hstmt, SQL_DROP);

        }
        else
        {
                car[0] = getc(fichier);
                while (car[0] != '1')
                {
                        car[0] = getc(fichier);
                }
                car[0] = getc(fichier);
                car[0] = getc(fichier);
                car[0] = getc(fichier);
                j=5;
```

```
                }
        }
        car[0] = getc(fichier);
        while (car[0] != '\n')
        {
                car[0] = getc(fichier);
        }
        car[0] = getc(fichier);
        while ((car[0] != '\n')&&(!feof(fichier)))
        {
                car[0] = getc(fichier);
        }
        car[0] = getc(fichier);

}


rc = SQLDisconnect(hdbc);
rc = SQLFreeConnect(hdbc);
rc = SQLFreeEnv(henv);

fclose(fichier);

};




void main()
{
InitialiseBd ();
}



MLOCAL BYTE * PNEAR BuildString(FILE * fichier)
{
LOCAL BYTE *pbuffer=NULL, car[2]=" ";
size_t size;
int count1=0;
int count2=0;

   /*
    * Allocate memory for character string buffer
    */
if( (pbuffer = (BYTE *)malloc( MAX_BUF_SIZE * sizeof( BYTE ) )) == NULL )
{
        /* Error */
        return(NULL);
```

```
}

size = _msize( pbuffer );

while ((car[0] = getc(fichier)) != '\n')
{

        /* Copy Character into String */
        pbuffer[(count1++)+(count2*MAX_BUF_SIZE)]= car[0];

        /*
                * Verify That There is still place in string to include terminating
characters,
                * Otherwise, reallocate MAX_BUF_SIZE more bytes to buffer.
                */
        if
        ((count1+(count2*MAX_BUF_SIZE))>((count2*MAX_BUF_SIZE)+(MAX_BUF_SI
ZE-2)))
        {

                count1=-1;
                count2++;

                /* Reallocate MAX_BUF_SIZE more bytes to Buffer */
                if( (pbuffer = (BYTE *)realloc( (void *)pbuffer, size + (MAX_BUF_SIZE *(
sizeof( BYTE )) ))) == NULL )
                {
                        /* Error */
                        return(NULL);
                }

                size = _msize( pbuffer );
        }

        /* Read Next Character */
}



/* Attach terminating Characters to string */
pbuffer[count1+(count2*MAX_BUF_SIZE)]='\0';

return(pbuffer);
}
```

Annexe 1 : Le code C de la DLL du dictionnaire multilingue

```
/*
 * file dico.c
 */


#include "sql.h"
#include "sqlext.h"

#define DatabaseName "Apollo"
#define TableSize 3000

/*Déclaration des fonctions*/

GLOBAL int  PFAR  TpsQueryServiceCount(void );
void    SelectionNumerosMotPremier (HWND hDlg);
void    SelectionDefinition (HWND hDlg);
void    SelectionNumerosMot (BYTE buffer [50], LONG int numero);
void    SelectionSynonymes (HWND hDlg, LONG int def);
void    SelectionTraductions (HWND hDlg, LONG int def);
void    RechercheBd(HWND hDlg);
void    RechercheSelection (HWND hDlg,LONG int numero);
BOOL FAR PASCAL _export MyDlgProc (HWND hDlg, UINT message, UINT wParam,
LONG lParam);
GLOBAL void PFAR TpsCallService(LONG lVal, WORD wService );
GLOBAL void PFAR TpsNotifyEvent(LONG lVal, WORD wEvent, LONG lParam1, LONG
lParam2 );
GLOBAL void PFAR TpsProc(WORD wSrvFunctionId, TS_MISC pSrvFunction );


/*Declaration de variables globales*/

MLOCAL     BYTE azMenuItem[] = "Spécial, Césure";
MLOCAL     FARPROC    lpfnDlgProc;
MLOCAL     BYTE azBuffer[255];
MLOCAL  BYTE     bufsource[50];
MLOCAL  BYTE     bufcible[50];
MLOCAL  BYTE     bufmot[255];
MLOCAL  BYTE     bufrempl[255];
MLOCAL  BYTE     langue[50];
MLOCAL  BYTE     ATraduire [1255];
MLOCAL  BYTE     bufinters [50];
MLOCAL  BYTE     bufinterc [50];

HENV henv;
HDBC        hdbc;
HSTMT       hstmt;
RETCODE rc;
```

```
SDWORD      cbvalue = SQL_NTS;

LONG int tNumMot [TableSize];
int NumMot = 0;
LONG int tNumDef [TableSize];
int NumDef = 0;

BOOL selection = FALSE;
BOOL          source = FALSE;
BOOL          cible = FALSE;



GLOBAL      int      PFAR  TpsQueryServiceCount(void )
{
        return(1 );
}



/*****sélection des numeros du mot entré par l'utilisateur*****/

void SelectionNumerosMotPremier (HWND hDlg)
{

LONG int numero ;
SDWORD      cbnumero;
LOCAL       BYTE param [255];
LOCAL       BYTE message[50];

NumMot = 0;

/*allocation d'un 'statement'*/
rc = SQLAllocStmt(hdbc,&hstmt);

/*préparation de la commande SQL1: sélection des numéros indentifiants du mot à
traduire*/
if (strcmp(bufsource,"Français") == 0)
        rc = SQLPrepare(hstmt,"SELECT num_mot FROM français WHERE MOT
                LIKE ? ",SQL_NTS);
else if (strcmp(bufsource,"Anglais") == 0)
        rc = SQLPrepare(hstmt,"SELECT num_mot FROM anglais WHERE MOT
                LIKE ?",SQL_NTS);
else if (strcmp(bufsource,"Allemand") == 0)
        rc = SQLPrepare(hstmt,"SELECT num_mot FROM allemand WHERE MOT
                LIKE ?",SQL_NTS);
else if (strcmp(bufsource,"Néerlandais") == 0)
        rc = SQLPrepare(hstmt,"SELECT num_mot FROM néerld WHERE MOT
                LIKE ?",SQL_NTS);

/*lien avec le paramètre 'bufmot'*/
```

```
strcpy(param,"%");
strcat(param,bufmot);
strcat(param,"%");
rc =
SQLBindParameter(hstmt,1,SQL_PARAM_INPUT_OUTPUT,SQL_C_DEFAULT,SQL_VA
RCHAR,255,255,param,255,&cbvalue) ;

/*exécution de la commande SQL1*/
rc = SQLExecute(hstmt);

if ((rc == SQL_SUCCESS) || (rc == SQL_SUCCESS_WITH_INFO))
{
        /*organisation des résultats*/
        rc = SQLBindCol(hstmt,1,SQL_C_SLONG,(PTR)&numero,0,&cbnumero);
        /*recherche des résultats numero*/
        while (TRUE)
        {
                rc = SQLFetch(hstmt);
                if (rc == SQL_SUCCESS)
                {
                        /*remplissage du taleau de numéro de mot*/
                        tNumMot[NumMot] = numero;
                        NumMot = NumMot+1;
                }
                else break;
        }
        /*le mot n'est pas dans la BD*/
        if (NumMot == 0)
        {
                strcpy(message,"Ce mot ");
                strcat(message,bufsource);
                strcat(message," ne figure pas dans le dictionaire.");
                MessageBox(hDlg,message,"Attention",MB_ICONEXCLAMATION);
        };
        SQLFreeStmt(hstmt,SQL_DROP);
}
/*erreur lors de la recherche dans la BD*/
else if (rc == SQL_ERROR)
        MessageBox(hDlg,"La base de données à laquelle vous êtes connecté n'a pas la bonne
                structure.","Attention",MB_ICONEXCLAMATION);
};


/*****sélection des numéros de définition + affichage def  et mot*****/

void SelectionDefinition (HWND hDlg)
{
int i;
LONG int ndef;
LONG int no,d;
SDWORD     cbndef;
```

```
SDWORD      cbdefi;
SDWORD      cbmottrouve;
LOCAL BYTE      defi[1000];
LOCAL BYTE      mottrouve [255];
LOCAL BYTE      liste[1265];
HSTMT       hstmt1;


NumDef = 0;

for (i=0;i<NumMot;i++)
{
        /*allocation d'un 'statement'*/
        rc = SQLAllocStmt(hdbc,&hstmt);

        no = tNumMot[i];

        /*préparation de la commande SQL2: sélection des numéros de définition des
mots à traduire*/
        if (strcmp(bufsource,"Français") == 0)
                rc = SQLPrepare(hstmt,"SELECT num_def FROM traduction_f
                        WHERE num_mot = ?",SQL_NTS);
        else if (strcmp(bufsource,"Anglais") == 0)
                rc = SQLPrepare(hstmt,"SELECT num_def FROM traduction_an
                        WHERE num_mot = ?",SQL_NTS);
        else if (strcmp(bufsource,"Allemand") == 0)
                rc = SQLPrepare(hstmt,"SELECT num_def FROM traduction_al
                        WHERE num_mot = ?",SQL_NTS);
        else if (strcmp(bufsource,"Néerlandais") == 0)
                rc = SQLPrepare(hstmt,"SELECT num_def FROM traduction_n
                        WHERE num_mot = ?",SQL_NTS);

        /*lien avec le paramètre no de mot*/
        rc =
        SQLBindParameter(hstmt,1,SQL_PARAM_INPUT_OUTPUT,SQL_C_SLONG,SQL
        _DOUBLE,0,0,(PTR)&no,0,&cbvalue) ;

        /*execution de la commande SQL2*/
        rc = SQLExecute(hstmt);
        if ((rc == SQL_SUCCESS) || (rc == SQL_SUCCESS_WITH_INFO))
        {
                /*organisation des résultats*/
                rc = SQLBindCol(hstmt,1,SQL_C_SLONG,(PTR)&ndef,0,&cbndef);
                /*recherche des résultats numéro de définition*/
                while (TRUE)
                {
                        rc = SQLFetch(hstmt);
                        if (rc == SQL_SUCCESS)
                        {
                                /*remplissage du taleau de numéro de def*/
                                tNumDef[NumDef]=ndef;
```

```
                        NumDef=NumDef+1;
                        rc = SQLAllocStmt(hdbc,&hstmt1);

                        if (strcmp(bufsource,"Français") == 0)
                        {
                                /*selon la langue dans laquelle on veut la définition*/
                                /*préparation de la requete SQL3:sélection du mot et
                                de la def correspondant a num-mot et num-def*/
                                if (strcmp(langue,"Français") == 0)
                                        rc = SQLPrepare(hstmt1,"SELECT
français.mot,definition.def_f FROM {oj français LEFT OUTER JOIN (traduction_f LEFT
OUTER JOIN definition ON traduction_f.num_def = definition.num_def) ON "
                                                        "français.num_mot =
traduction_f.num_mot} WHERE (traduction_f.num_def = ?) AND (traduction_f.num_mot=
?)",SQL_NTS);
                                else if (strcmp(langue,"Anglais") == 0)
                                        rc = SQLPrepare(hstmt1,"SELECT
français.mot,definition.def_an FROM {oj français LEFT OUTER JOIN (traduction_f LEFT
OUTER JOIN definition ON traduction_f.num_def = definition.num_def) ON "
                                                        "français.num_mot =
traduction_f.num_mot} WHERE (traduction_f.num_def = ?) AND (traduction_f.num_mot=
?)",SQL_NTS);
                                else if (strcmp(langue,"Allemand") == 0)
                                        rc = SQLPrepare(hstmt1,"SELECT
français.mot,definition.def_al FROM {oj français LEFT OUTER JOIN (traduction_f LEFT
OUTER JOIN definition ON traduction_f.num_def = definition.num_def) ON "
                                                        "français.num_mot =
traduction_f.num_mot} WHERE (traduction_f.num_def = ?) AND (traduction_f.num_mot=
?)",SQL_NTS);
                                else if (strcmp(langue,"Néerlandais") == 0)
                                        rc = SQLPrepare(hstmt1,"SELECT
français.mot,definition.def_n FROM {oj français LEFT OUTER JOIN (traduction_f LEFT
OUTER JOIN definition ON traduction_f.num_def = definition.num_def) ON "
                                                        "français.num_mot =
traduction_f.num_mot} WHERE (traduction_f.num_def = ?) AND (traduction_f.num_mot=
?)",SQL_NTS);
                        }
                        else if (strcmp(bufsource,"Anglais") == 0)
                        {
                                if (strcmp(langue,"Français") == 0)
                                        rc = SQLPrepare(hstmt1,"SELECT
anglais.mot,definition.def_f FROM {oj anglais LEFT OUTER JOIN (traduction_an LEFT
OUTER JOIN definition ON traduction_an.num_def = definition.num_def) ON "
                                                        "anglais.num_mot =
traduction_an.num_mot} WHERE (traduction_an.num_def = ?) AND
(traduction_an.num_mot= ?)",SQL_NTS);
                                else if (strcmp(langue,"Anglais") == 0)

                                        rc = SQLPrepare(hstmt1,"SELECT
anglais.mot,definition.def_an FROM {oj anglais LEFT OUTER JOIN (traduction_an LEFT
OUTER JOIN definition ON traduction_an.num_def = definition.num_def) ON "
```

```
                                                "anglais.num_mot =
traduction_an.num_mot} WHERE (traduction_an.num_def = ?) AND
(traduction_an.num_mot= ?)",SQL_NTS);
                                        else if (strcmp(langue,"Allemand") == 0)
                                                rc = SQLPrepare(hstmt1,"SELECT
anglais.mot,definition.def_al FROM {oj anglais LEFT OUTER JOIN (traduction_an LEFT
OUTER JOIN definition ON traduction_an.num_def = definition.num_def) ON "
                                                "anglais.num_mot =
traduction_an.num_mot} WHERE (traduction_an.num_def = ?) AND
(traduction_an.num_mot= ?)",SQL_NTS);
                                        else if (strcmp(langue,"Néerlandais") == 0)
                                                rc = SQLPrepare(hstmt1,"SELECT
anglais.mot,definition.def_n FROM {oj anglais LEFT OUTER JOIN (traduction_an LEFT
OUTER JOIN definition ON traduction_an.num_def = definition.num_def) ON "
                                                "anglais.num_mot =
traduction_an.num_mot} WHERE (traduction_an.num_def = ?) AND
(traduction_an.num_mot= ?)",SQL_NTS);
                                }
                                else if (strcmp(bufsource,"Allemand") == 0)
                                {
                                        if (strcmp(langue,"Français") == 0)
                                                rc = SQLPrepare(hstmt1,"SELECT
allemand.mot,definition.def_f FROM {oj allemand LEFT OUTER JOIN (traduction_al LEFT
OUTER JOIN definition ON traduction_al.num_def = definition.num_def) ON "
                                                "allemand.num_mot =
traduction_al.num_mot} WHERE (traduction_al.num_def = ?) AND (traduction_al.num_mot=
?)",SQL_NTS);
                                        else if (strcmp(langue,"Anglais") == 0)
                                                rc = SQLPrepare(hstmt1,"SELECT
allemand.mot,definition.def_an FROM {oj allemand LEFT OUTER JOIN (traduction_al LEFT
OUTER JOIN definition ON traduction_al.num_def = definition.num_def) ON "
                                                "allemand.num_mot =
traduction_al.num_mot} WHERE (traduction_al.num_def = ?) AND (traduction_al.num_mot=
?)",SQL_NTS);
                                        else if (strcmp(langue,"Allemand") == 0)
                                                rc = SQLPrepare(hstmt1,"SELECT
allemand.mot,definition.def_al FROM {oj allemand LEFT OUTER JOIN (traduction_al LEFT
OUTER JOIN definition ON traduction_al.num_def = definition.num_def) ON "
                                                "allemand.num_mot =
traduction_al.num_mot} WHERE (traduction_al.num_def = ?) AND (traduction_al.num_mot=
?)",SQL_NTS);
                                        else if (strcmp(langue,"Néerlandais") == 0)
                                                rc = SQLPrepare(hstmt1,"SELECT
allemand.mot,definition.def_n FROM {oj allemand LEFT OUTER JOIN (traduction_al LEFT
OUTER JOIN definition ON traduction_al.num_def = definition.num_def) ON "
                                                "allemand.num_mot =
traduction_al.num_mot} WHERE (traduction_al.num_def = ?) AND (traduction_al.num_mot=
?)",SQL_NTS);
                                }
                                else if (strcmp(bufsource,"Néerlandais") == 0)
                                {
```

```
                        if (strcmp(langue,"Français") == 0)
                                rc = SQLPrepare(hstmt1,"SELECT
néerld.mot,definition.def_f FROM {oj néerld LEFT OUTER JOIN (traduction_n LEFT
OUTER JOIN definition ON traduction_n.num_def = definition.num_def) ON "
                                                "néerld.num_mot =
traduction_n.num_mot} WHERE (traduction_n.num_def = ?) AND (traduction_n.num_mot=
?)",SQL_NTS);
                        else if (strcmp(langue,"Anglais") == 0)
                                rc = SQLPrepare(hstmt1,"SELECT
néerld.mot,definition.def_an FROM {oj néerld LEFT OUTER JOIN (traduction_n LEFT
OUTER JOIN definition ON traduction_n.num_def = definition.num_def) ON "
                                                "néerld.num_mot =
traduction_n.num_mot} WHERE (traduction_n.num_def = ?) AND (traduction_n.num_mot=
?)",SQL_NTS);
                        else if (strcmp(langue,"Allemand") == 0)
                                rc = SQLPrepare(hstmt1,"SELECT
néerld.mot,definition.def_al FROM {oj néerld LEFT OUTER JOIN (traduction_n LEFT
OUTER JOIN definition ON traduction_n.num_def = definition.num_def) ON "
                                                "néerld.num_mot =
traduction_n.num_mot} WHERE (traduction_n.num_def = ?) AND (traduction_n.num_mot=
?)",SQL_NTS);
                        else if (strcmp(langue,"Néerlandais") == 0)
                                rc = SQLPrepare(hstmt1,"SELECT
néerld.mot,definition.def_n FROM {oj néerld LEFT OUTER JOIN (traduction_n LEFT
OUTER JOIN definition ON traduction_n.num_def = definition.num_def) ON "
                                                "néerld.num_mot =
traduction_n.num_mot} WHERE (traduction_n.num_def = ?) AND (traduction_n.num_mot=
?)",SQL_NTS);
                }

                /*lien avec les paramètres no de définition et no de mot*/
                rc =
SQLBindParameter(hstmt1,1,SQL_PARAM_INPUT_OUTPUT,SQL_C_SLONG,SQL_DOU
BLE,0,0,(PTR)&ndef,0,&cbvalue) ;
                rc =
SQLBindParameter(hstmt1,2,SQL_PARAM_INPUT_OUTPUT,SQL_C_SLONG,SQL_DOU
BLE,0,0,(PTR)&no,0,&cbvalue) ;

                /*execution de SQL3*/
                rc = SQLExecute(hstmt1);
                if ((rc == SQL_SUCCESS) || (rc ==
SQL_SUCCESS_WITH_INFO))
                {
                        /*organisation des résultats*/
                        rc =
SQLBindCol(hstmt1,1,SQL_C_CHAR,mottrouve,255,&cbmottrouve);
                        rc =
SQLBindCol(hstmt1,2,SQL_C_CHAR,defi,1000,&cbdefi);
                        /*recherche des résultats numéro*/
                        while (TRUE)
                        {
```

```
                                        rc = SQLFetch(hstmt1);
                                        if (rc == SQL_SUCCESS)
                                        {
                                                /*affichage des mots + définition*/
                                                strcpy(liste,"");
                                                strcpy(liste,mottrouve);
                                                strcat(liste,", ");
                                                strcat(liste,defi);

        SendDlgItemMessage(hDlg,IDC_LIST,LB_SETHORIZONTALEXTENT,4000,0L);
                                                        d =
SendDlgItemMessage(hDlg,IDC_LIST,LB_ADDSTRING,0,(LPARAM)((LPSTR)liste));
                                                /*on associe le numéro de def a chaque
item de la list box*/
                                                SendDlgItemMessage
(hDlg,IDC_LIST,LB_SETITEMDATA,(WPARAM)d,MAKELONG(ndef,0));

                                        }
                                        else break;
                                }
                        }
                        SQLFreeStmt(hstmt1,SQL_DROP);
                }
                else break;
        }
    }
    SQLFreeStmt(hstmt,SQL_DROP);
};
};
```

/***sélection des numéros de mot *****/**

```
void SelectionNumerosMot (BYTE buffer [50],LONG int numero)
{
LONG int no;
LONG int nmot ;
SDWORD cbnmot;


NumMot = 0;

/*allocation d'un 'statement'*/
rc = SQLAllocStmt(hdbc,&hstmt);

no = numero;

/*préparation de la commande SQL4: sélection des numéros de mots */
if (strcmp(buffer,"Français") == 0)
```

```
        rc = SQLPrepare(hstmt,"SELECT num_mot FROM traduction_f WHERE num_def =
                ?",SQL_NTS);
else if (strcmp(buffer,"Anglais") == 0)
        rc = SQLPrepare(hstmt,"SELECT num_mot FROM traduction_an WHERE num_def
                = ?",SQL_NTS);
else if (strcmp(buffer,"Allemand") == 0)
        rc = SQLPrepare(hstmt,"SELECT num_mot FROM traduction_al WHERE num_def =
                ?",SQL_NTS);
else if (strcmp(buffer,"Néerlandais") == 0)
        rc = SQLPrepare(hstmt,"SELECT num_mot FROM traduction_n WHERE num_def =
                ?",SQL_NTS);

/*lien avec le paramètre no def*/
rc =
SQLBindParameter(hstmt,1,SQL_PARAM_INPUT_OUTPUT,SQL_C_SLONG,SQL_DOUB
LE,0,0,(PTR)&no,0,&cbvalue) ;

/*execution de la commande SQL4*/
rc = SQLExecute(hstmt);
if ((rc == SQL_SUCCESS) || (rc == SQL_SUCCESS_WITH_INFO))
{
        /*organisation des résultats*/
        rc = SQLBindCol(hstmt,1,SQL_C_SLONG,(PTR)&nmot,0,&cbnmot);
        /*recherche des résultats numéro de mot*/
        while (TRUE)
        {
                rc = SQLFetch(hstmt);
                if (rc == SQL_SUCCESS)
                {
                        /*remplissage du tableau de numéro de mot*/
                        tNumMot[NumMot]=nmot;
                        NumMot=NumMot+1;
                }
                else break;
        }
}
SQLFreeStmt(hstmt,SQL_DROP);

};


/*****sélection des synonymes*****/

void SelectionSynonymes (HWND hDlg, LONG int def)
{
LOCAL BYTE      syno [255];
SDWORD          cbsyno;
LOCAL BYTE      ge [32];
SDWORD          cbge;
LOCAL BYTE      ca [32];
SDWORD          cbca;
```

```
LOCAL BYTE      co [255];
SDWORD          cbco;
LOCAL BYTE      res [255];
SDWORD          cbres;
LOCAL BYTE      ex [500];
SDWORD          cbex;
LOCAL BYTE      temp [1250];
int i;
LONG int no;
DWORD     dwIndex;


/*pour tout mot*/
for(i=0;i<NumMot;i++)
{
        rc = SQLAllocStmt(hdbc,&hstmt);

        no = tNumMot[i];

        /*préparation de SQL5: sélection des synonymes du mot*/
        if (strcmp(bufsource,"Français") == 0)
                rc = SQLPrepare(hstmt,
"SELECT
français.mot,français.categorie,français.genre,traduction_f.restriction,traduction_f.comment,tra
duction_f.exemple FROM {oj français LEFT OUTER JOIN traduction_f on français.num_mot
= traduction_f.num_mot} WHERE (français.num_mot=?) AND"
"(traduction_f.num_def=?)",SQL_NTS);
        else if (strcmp(bufsource,"Anglais") == 0)
                rc = SQLPrepare(hstmt,
"SELECT
anglais.mot,anglais.categorie,anglais.ae_be,traduction_an.restriction,traduction_an.comment,tr
aduction_an.exemple FROM {oj anglais LEFT OUTER JOIN traduction_an on
anglais.num_mot = traduction_an.num_mot } WHERE (anglais.num_mot=?)AND"
"(traduction_an.num_def=?)",SQL_NTS);
        else if (strcmp(bufsource,"Allemand") == 0)
                rc = SQLPrepare(hstmt,
"SELECT
allemand.mot,allemand.categorie,allemand.genre,traduction_al.restriction,traduction_al.comme
nt,traduction_al.exemple FROM {oj allemand LEFT OUTER JOIN traduction_al on
allemand.num_mot=traduction_al.num_mot} WHERE (allemand.num_mot=?) AND"
"(traduction_al.num_def=?)",SQL_NTS);
        else if (strcmp(bufsource,"Néerlandais") == 0)
                rc = SQLPrepare(hstmt,
"SELECT
néerld.mot,néerld.categorie,néerld.genre,traduction_n.restriction,traduction_n.comment,traduc
tion_n.exemple FROM {oj néerld LEFT OUTER JOIN traduction_n on
néerld.num_mot=traduction_n.num_mot} WHERE (néerld.num_mot=?) AND"
"(traduction_n.num_def=?)",SQL_NTS);

        /*lien des paramètres no mot et no def*/
```

```
        rc =
SQLBindParameter(hstmt,1,SQL_PARAM_INPUT_OUTPUT,SQL_C_LONG,SQL_DOUBL
E,0,0,(PTR)&no,0,&cbvalue) ;
        rc =
SQLBindParameter(hstmt,2,SQL_PARAM_INPUT_OUTPUT,SQL_C_LONG,SQL_DOUBL
E,0,0,(PTR)&def,0,&cbvalue) ;

        /*execution de SQL5*/
        rc = SQLExecute(hstmt);
        if ((rc == SQL_SUCCESS) || (rc == SQL_SUCCESS_WITH_INFO))
        {
                /*organisation des résultats*/
                rc = SQLBindCol(hstmt,1,SQL_C_CHAR,syno,255,&cbsyno);
                rc = SQLBindCol(hstmt,2,SQL_C_CHAR,ca,32,&cbca);
                rc = SQLBindCol(hstmt,3,SQL_C_CHAR,ge,32,&cbge);
                rc = SQLBindCol(hstmt,4,SQL_C_CHAR,res,225,&cbres);
                rc = SQLBindCol(hstmt,5,SQL_C_CHAR,co,225,&cbco);
                 rc = SQLBindCol(hstmt,6,SQL_C_CHAR,ex,500,&cbex);

                /*recherche des résultats synonymes*/
                while (TRUE)
                {
                        rc = SQLFetch(hstmt);
                        if (rc == SQL_SUCCESS)
                        {
                                /*mise en ligne des différents synonymes*/
                                strcpy(temp,"");
                                strcat (temp,syno);
                                if ((strcmp(ca,"cat") != 0) && (strcmp(ca,"") != 0))
                                {
                                        strcat (temp,", ");
                                        strcat (temp,ca);
                                };
                                if ((strcmp(ge,"gen") != 0) && (strcmp(ge,"be_vs_ae") != 0) &&
                                        (strcmp(ge,"") != 0))
                                {
                                        strcat (temp,", ");
                                        strcat (temp,ge);
                                };
                                if ((strcmp(res,"restriction") != 0) && (strcmp(res,"") != 0) &&
                                        (strcmp(res,"znl_vs_nnl") != 0))
                                {
                                        strcat (temp,", ");
                                        strcat (temp,res);
                                };
                                if ((strcmp(co,"comment") != 0) && (strcmp(co,"") != 0))
                                {
                                        strcat (temp,", ");
                                        strcat (temp,co);
                                };
                                if (strcmp(ex,"") != 0)
```

```
                                {
                                        strcat (temp,", ");
                                        strcat (temp,ex);
                                };
                                strcat (temp,", ");
                                /*affichage ds la dialogbox*/

        SendDlgItemMessage(hDlg,IDC_SYN,LB_SETHORIZONTALEXTENT,2500,0L);
                                        dwIndex =
SendDlgItemMessage(hDlg,IDC_SYN,LB_FINDSTRING,0,(LPARAM)((LPSTR)temp));
                                        if (dwIndex == LB_ERR)

        SendDlgItemMessage(hDlg,IDC_SYN,LB_ADDSTRING,0,(LPARAM)((LPSTR)tem
p));

                        }
                        else break;
                }

        };
        SQLFreeStmt(hstmt,SQL_DROP);
}

}


/*****recherche des traductions*****/

void SelectionTraductions (HWND hDlg, LONG int def)
{
LOCAL BYTE  message [50];
LOCAL BYTE      ge [32];
SDWORD          cbge;
LOCAL BYTE      ca [32];
SDWORD          cbca;
LOCAL BYTE      co [255];
SDWORD          cbco;
LOCAL BYTE      res [255];
SDWORD          cbres;
LOCAL BYTE      ex [500];
SDWORD          cbex;
LOCAL BYTE      trad [255];
SDWORD          cbtrad;
LOCAL BYTE      temp2 [1250];
int i = 0;
int j = 0;
int k = 0;
LONG int no;
DWORD       dwIndex;
```

```
/*pour tout mot*/
for(i=0;i<NumMot;i++)
{

        rc = SQLAllocStmt(hdbc,&hstmt);

        no = tNumMot[i] ;
        /*préparation SQL6: recherche des traductions du mot*/
        if (strcmp(bufcible,"Français") == 0)
                rc = SQLPrepare(hstmt,
"SELECT
français.mot,français.categorie,français.genre,traduction_f.restriction,traduction_f.comment,tra
duction_f.exemple FROM {oj français LEFT OUTER JOIN traduction_f on français.num_mot
= traduction_f.num_mot } WHERE (traduction_f.num_mot=?) AND"
"(traduction_f.num_def=?)" ,SQL_NTS);
        else if (strcmp(bufcible,"Anglais") == 0)
                rc = SQLPrepare(hstmt,
"SELECT
anglais.mot,anglais.categorie,anglais.ae_be,traduction_an.restriction,traduction_an.comment,tr
aduction_an.exemple FROM {oj anglais LEFT OUTER JOIN traduction_an on
anglais.num_mot = traduction_an.num_mot } WHERE (anglais.num_mot=?) AND"
"(traduction_an.num_def=?)",SQL_NTS);
        else if (strcmp(bufcible,"Allemand") == 0)
                rc = SQLPrepare(hstmt,
"SELECT
allemand.mot,allemand.categorie,allemand.genre,traduction_al.restriction,traduction_al.comme
nt,traduction_al.exemple FROM {oj allemand LEFT OUTER JOIN traduction_al on
allemand.num_mot=traduction_al.num_mot} WHERE (allemand.num_mot=?) AND"
"(traduction_al.num_def=?)",SQL_NTS);
        else if (strcmp(bufcible,"Néerlandais") == 0)
                rc = SQLPrepare(hstmt,
"SELECT
néerld.mot,néerld.categorie,néerld.genre,traduction_n.restriction,traduction_n.comment,traduc
tion_n.exemple FROM {oj néerld LEFT OUTER JOIN traduction_n on
néerld.num_mot=traduction_n.num_mot} WHERE (néerld.num_mot=?) AND"
"(traduction_n.num_def=?)",SQL_NTS);


        /*lien des paramètres num de mot et no def*/
        rc =
SQLBindParameter(hstmt,1,SQL_PARAM_INPUT_OUTPUT,SQL_C_LONG,SQL_DOUBL
E,0,0,(PTR)&no,0,&cbvalue) ;
        rc =
SQLBindParameter(hstmt,2,SQL_PARAM_INPUT_OUTPUT,SQL_C_LONG,SQL_DOUBL
E,0,0,(PTR)&def,0,&cbvalue) ;

        /*execution de SQL6*/
        rc = SQLExecute(hstmt);
        if ((rc == SQL_SUCCESS) || (rc == SQL_SUCCESS_WITH_INFO))
        {
                /*organisation des résultats*/
```

```
rc = SQLBindCol(hstmt,1,SQL_C_CHAR,trad,255,&cbtrad);
rc = SQLBindCol(hstmt,2,SQL_C_CHAR,ca,32,&cbca);
rc = SQLBindCol(hstmt,3,SQL_C_CHAR,ge,32,&cbge);
rc = SQLBindCol(hstmt,4,SQL_C_CHAR,res,255,&cbres);
rc = SQLBindCol(hstmt,5,SQL_C_CHAR,co,255,&cbco);
rc = SQLBindCol(hstmt,6,SQL_C_CHAR,ex,500,&cbex);


/*recherche des résultats traductions*/
while (TRUE)
{
        rc = SQLFetch(hstmt);

        if (rc == SQL_SUCCESS)
        {
                k=k+1;
                /*mise en ligne des différentes traductions*/
                strcpy(temp2,"");
                strcat (temp2,trad);
                if ((strcmp(ca,"cat") != 0) && (strcmp(ca,"") != 0))
                {
                        strcat (temp2,", ");
                        strcat (temp2,ca);
                };
                if ((strcmp(ge,"gen") != 0) && (strcmp(ge,"be_vs_ae") != 0) &&
                        (strcmp(ge,"") != 0))
                {
                        strcat (temp2,", ");
                        strcat (temp2,ge);
                };
                if ((strcmp(res,"restriction") != 0) && (strcmp(res,"") != 0) &&
                        (strcmp(res,"znl_vs_nnl") != 0))
                {
                        strcat (temp2,", ");
                        strcat (temp2,res);
                };
                if ((strcmp(co,"comment") != 0) && (strcmp(co,"") != 0))
                {
                        strcat (temp2,", ");
                        strcat (temp2,co);
                };
                if (strcmp(ex,"") != 0)
                {
                        strcat (temp2,", ");
                        strcat (temp2,ex);
                };
                strcat (temp2,", ");
                /*affichage ds la dialogbox*/

SendDlgItemMessage(hDlg,IDC_TRAD,LB_SETHORIZONTALEXTENT,2500,0L);
```

```
                                        dwIndex =
SendDlgItemMessage(hDlg,IDC_TRAD,LB_FINDSTRING,0,(LPARAM)((LPSTR)temp2));
                                        if (dwIndex == LB_ERR)

        SendDlgItemMessage(hDlg,IDC_TRAD,LB_ADDSTRING,0,(LPARAM)((LPSTR)te
mp2));
                        }
                        else break;
                }

        };
        SQLFreeStmt(hstmt,SQL_DROP);

}
if (k == 0)
{
        strcpy(message,"Aucune traduction en ");
        strcat(message,bufcible);
        strcat(message," n'a été trouvée.");
        MessageBox(hDlg,message,"Attention",MB_ICONEXCLAMATION);
};
}
```

/***recherche dans la BD*****/**

```
void RechercheBd(HWND hDlg)
{
```
/*remise a blanc des listebox contenant trad et syn et list et du champ de définition*/
```
SendDlgItemMessage(hDlg,IDC_LIST,LB_RESETCONTENT,0,0);
SendDlgItemMessage(hDlg,IDC_SYN,LB_RESETCONTENT,0,0);
SendDlgItemMessage(hDlg,IDC_TRAD,LB_RESETCONTENT,0,0);
SetDlgItemText(hDlg,IDC_DEF,"");
```

/*sélection des numéros correspondant au mot de l'utilisateur*/
```
SelectionNumerosMotPremier(hDlg);
```
/*sélection des numéros de definition correspondants a ces num_mot + affichage du mot et sa def dans list*/
```
SelectionDefinition (hDlg);
}
```

/***arrangement avant traduction*****/**

```
void RechercheSelection (HWND hDlg, LONG int numero)
{
LOCAL BYTE *pb, *pb2;
LOCAL BYTE *pbMot ;
LOCAL BYTE *pbDef ;
```

/*mise a blanc des list box syn et trad*/

```
SendDlgItemMessage(hDlg,IDC_SYN,LB_RESETCONTENT,0,0);
SendDlgItemMessage(hDlg,IDC_TRAD,LB_RESETCONTENT,0,0);

/*affichage de la définition a partir de la sélection de l'utilisateur*/
pbMot=(BYTE *)malloc(sizeof(BYTE)*(strlen(ATraduire)+1));
strcpy(pbMot,ATraduire);
pb = strchr(pbMot,',');
pb2=pb+2;
pbDef=(BYTE *)malloc(sizeof(BYTE)*(strlen(pb2)+1));
strcpy(pbDef,pb2);
*pb=0;

SetDlgItemText(hDlg,IDC_DEF,pbDef);

/*sélection des num mot en langue source correspondant au no def*/
SelectionNumerosMot (bufsource,numero);
/*sélection des synonymes*/
SelectionSynonymes(hDlg,numero);
/*sélection des num mot en langue cible correspondant au no def*/
SelectionNumerosMot (bufcible,numero);
/*sélection des traductions*/
SelectionTraductions(hDlg,numero);

free(pbMot);
free(pbDef);

}


/*****boite de dialogue *****/

BOOL FAR PASCAL _export MyDlgProc (HWND hDlg, UINT message, UINT wParam,
LONG lParam)
{
int checked;
LONG int numero;
DWORD d,nIndex;
BYTE * pb;
HCURSOR hCursor;

switch(message)
{
        case WM_INITDIALOG:

        strcpy(bufrempl,"");
        /*initialisation des combos*/
        /*langues sources*/

        SendDlgItemMessage(hDlg,IDC_LANGUES,CB_ADDSTRING,0,(LPARAM)((LPST
R)"Anglais"));
```

```
        SendDlgItemMessage(hDlg,IDC_LANGUES,CB_ADDSTRING,0,(LPARAM)((LPST
R)"Français"));

        SendDlgItemMessage(hDlg,IDC_LANGUES,CB_ADDSTRING,0,(LPARAM)((LPST
R)"Allemand"));

        SendDlgItemMessage(hDlg,IDC_LANGUES,CB_ADDSTRING,0,(LPARAM)((LPST
R)"Néerlandais"));

        /*langues cibles*/

        SendDlgItemMessage(hDlg,IDC_LANGUEC,CB_ADDSTRING,0,(LPARAM)((LPS
TR)"Anglais"));

        SendDlgItemMessage(hDlg,IDC_LANGUEC,CB_ADDSTRING,0,(LPARAM)((LPS
TR)"Français"));

        SendDlgItemMessage(hDlg,IDC_LANGUEC,CB_ADDSTRING,0,(LPARAM)((LPS
TR)"Allemand"));

        SendDlgItemMessage(hDlg,IDC_LANGUEC,CB_ADDSTRING,0,(LPARAM)((LPS
TR)"Néerlandais"));

        /*initialisation du mot à traduire*/
        if (strcmp(bufmot,"")!=0)
                SetDlgItemText(hDlg,IDC_EDIT1,bufmot);
        else
                SetDlgItemText(hDlg,IDC_EDIT1,azBuffer);

        /*affichage du dernier choix de l'utilisateur pour ls et lc + initialisation des entete
de SYN et TRAD*/
        nIndex =
        SendDlgItemMessage(hDlg,IDC_LANGUES,CB_SELECTSTRING,(WPARAM)-
1,(LPARAM)((LPSTR)bufsource));

        SendDlgItemMessage(hDlg,IDC_LANGUES,CB_SETCURSEL,(WPARAM)nIndex,0
);
        SetDlgItemText(hDlg,IDC_TRADS,bufsource);
                        nIndex =
SendDlgItemMessage(hDlg,IDC_LANGUEC,CB_SELECTSTRING,(WPARAM)-
1,(LPARAM)((LPSTR)bufcible));

        SendDlgItemMessage(hDlg,IDC_LANGUEC,CB_SETCURSEL,(WPARAM)nIndex,0
);
        SetDlgItemText(hDlg,IDC_TRADC,bufcible);

        /*initialisation du bouton radio*/
        if(strcmp(langue,"") == 0)
```

```
        SendDlgItemMessage(hDlg,IDC_RADIO1,BM_SETCHECK,1,0);

else if (strcmp(langue,bufsource) == 0)

        SendDlgItemMessage(hDlg,IDC_RADIO1,BM_SETCHECK,1,0);
else if (strcmp(langue,bufcible) == 0)

        SendDlgItemMessage(hDlg,IDC_RADIO2,BM_SETCHECK,1,0);

/*affichage des infos si ls, lc et mot initialise avant ouverture de la fenetre*/
if ((strcmp(bufsource,"") != 0) && (strcmp(bufcible,"") != 0) && (strcmp(azBuffer,"")
        != 0))
{
        /*curseur devient sablier*/
        hCursor = SetCursor (LoadCursor((HINSTANCE)NULL,IDC_WAIT));
        ShowCursor(TRUE);

        /*appel a une fonction de recherche dans la BD*/
        RechercheBd(hDlg);

        /*curseur redevient fleche*/
        ShowCursor(FALSE);
        SetCursor(hCursor);
};
return TRUE;
break;

case WM_COMMAND:
switch(wParam)
{
        case IDOK:
        strcpy(bufrempl,"");
        /*message d'erreur*/
        /*pas de mot*/
        if (strcmp(bufmot,"") == 0)
        {
                MessageBox(hDlg,"Vous n'avez pas precisé le mot à
                traduire.","Attention",MB_ICONEXCLAMATION);
                return 0;
                strcpy(azBuffer,"");
        }
        /*pas de langue source*/
        else if (strcmp(bufsource,"") == 0)
        {
                MessageBox(hDlg,"Vous n'avez pas precisé la langue
                source.","Attention",MB_ICONEXCLAMATION );
                return 0;
        }
        /* pas de langue cible*/
        else if (strcmp(bufcible,"") == 0)
```

```
{
        MessageBox(hDlg,"Vous n'avez pas precisé la langue
        cible.","Attention",MB_ICONEXCLAMATION);
        return 0;
}
/* langue source=langue cible*/
if ( (strcmp(bufsource,bufcible) == 0) && (strcmp(bufsource,"") != 0) )
{
        MessageBox(hDlg,"La langue source est indentique à la langue
        cible.","Attention",MB_ICONEXCLAMATION);
        return 0;
}

if ((strcmp(bufsource,"") != 0) && (strcmp(bufmot,"") != 0))
{
        strcpy(azBuffer,bufmot);

        /*initialisation de la langue d'affichage des définitions*/
        if (strcmp(langue,"") == 0)
                strcpy(langue,bufsource);

        if ((strcmp(bufinters,bufsource) != 0) && (source == TRUE))
                strcpy(langue,bufsource);

        else if ((strcmp(bufinterc,bufcible) != 0) && (cible == TRUE))
                strcpy(langue,bufcible);

        /*curseur devient sablier*/
        hCursor = SetCursor
                (LoadCursor((HINSTANCE)NULL,IDC_WAIT));
        ShowCursor(TRUE);

        /*appel a une fonction de recherche dans la BD*/
        RechercheBd(hDlg);

        /*curseur redevient fleche*/
        ShowCursor(FALSE);
        SetCursor(hCursor);

        return TRUE;
}
break;

case IDCANCEL:

/*préparation de l'insertion à effectuer dans interscript*/
if (strcmp(bufrempl,"") != 0)
{
        pb = strchr(bufrempl,',');
        *pb = 0;
}
```

```
                /*fermeture de la boite*/
                EndDialog(hDlg,0);
                /*deconnexion de la BD*/
                SQLDisconnect(hdbc);
                SQLFreeConnect(hdbc);
                SQLFreeEnv(henv);
                return FALSE;
                break;

                case IDC_mot:
                /*saisie du mot*/
                GetDlgItemText(hDlg,IDC_mot,bufmot,50);
                break;

                case IDC_LANGUES:
                if (HIWORD (lParam) == LBN_SELCHANGE)
                {
                        strcpy(bufinters,bufsource);
                        /*saisie dans la combo de la langue source*/
                        d =
SendDlgItemMessage(hDlg,IDC_LANGUES,CB_GETCURSEL,0,0);

        SendDlgItemMessage(hDlg,IDC_LANGUES,CB_GETLBTEXT,(WPARAM)d,(LPA
RAM)((LPSTR) bufsource));
                        /*initialisation de l'entete de la liste des synonymes*/
                        SetDlgItemText(hDlg,IDC_TRADS,bufsource);
                };
                break;

                case IDC_LANGUEC:
                if (HIWORD (lParam) == LBN_SELCHANGE)
                {
                        strcpy(bufinterc,bufcible);
                        /*saisie dans la combo de la langue source*/
                        d =
SendDlgItemMessage(hDlg,IDC_LANGUEC,CB_GETCURSEL,0,0);

        SendDlgItemMessage(hDlg,IDC_LANGUEC,CB_GETLBTEXT,(WPARAM)
d,(LPARAM)((LPSTR) bufcible));
                        /*initialisation de l'entete de la liste des traductions*/
                        SetDlgItemText(hDlg,IDC_TRADC,bufcible);
                };
                break;

                case IDC_RADIO1:
                source = TRUE;
                cible = FALSE;
                checked = (int)
                SendDlgItemMessage(hDlg,IDC_RADIO1,BM_GETCHECK,0,0);
                if (checked == 1)
                        strcpy(langue,bufsource);
```

```
                break;

                case IDC_RADIO2:
                cible = TRUE;
                source = FALSE;
                checked =(int)
                SendDlgItemMessage(hDlg,IDC_RADIO2,BM_GETCHECK,0,0);
                if (checked == 1)
                strcpy(langue,bufcible);

                break;

                case IDC_LIST:
                /*appel de traduction lors de la sélection dans list*/
                if (HIWORD (lParam) == LBN_SELCHANGE)
                {
                        d = SendDlgItemMessage(hDlg,IDC_LIST,LB_GETCURSEL,0,0);
                        /*on retrouve le numéro de def associe a la sélection dans la list
                                box*/
                        numero =
SendDlgItemMessage(hDlg,IDC_LIST,LB_GETITEMDATA,(WPARAM)d,0);


        SendDlgItemMessage(hDlg,IDC_LIST,LB_GETTEXT,(WPARAM)
d,(LPARAM)((LPSTR) ATraduire));
                        RechercheSelection(hDlg,numero);
                };
                break;

                case IDC_SYN:
                d = SendDlgItemMessage(hDlg,IDC_SYN,LB_GETCURSEL,0,0);

                SendDlgItemMessage(hDlg,IDC_SYN,LB_GETTEXT,(WPARAM)
d,(LPARAM)((LPSTR) bufrempl));
                break;

                case IDC_TRAD:
                d = SendDlgItemMessage(hDlg,IDC_TRAD,LB_GETCURSEL,0,0);
                                SendDlgItemMessage(hDlg,IDC_TRAD,LB_GETTEXT,
(WPARAM)d,(LPARAM)((LPSTR) bufrempl));

                default :
                break;
        }
        break;
};
return FALSE;

}
```

```
GLOBAL      void    PFAR  TpsCallService(LONG lVal, WORD wService )
{

LOCAL       WORD        iPos;
LOCAL BYTE selection[50];

switch (wService )
{
        case 1  :

                /* save current position */
                iPos = (*pTpsSavePosition)(lVal, TRUE );
                /* test selection or select word */
                (*pTpsQuerySelection)(lVal, azBuffer, 255);
                strcpy(selection,azBuffer);
                /*connexion avec la BD*/
                rc = SQLAllocEnv(&henv);
                rc = SQLAllocConnect(henv,&hdbc);
                rc = SQLConnect(hdbc,DatabaseName,SQL_NTS,"" ,SQL_NTS,""
                        ,SQL_NTS);

                if (rc != SQL_SUCCESS)
                        (*pTpsOkBox)(lVal,"Mauvaise configuration ODBC: la base de
                        données doit être du type 'apollo (Microsoft Access Driver)'");
                else
                {
                        /*appel de la boite de dialogue */
                        lpfnDlgProc = MakeProcInstance((FARPROC)MyDlgProc,
                                hTpsInstance);
                        DialogBox(hTpsInstance, MAKEINTRESOURCE(101),
                                GetActiveWindow(), lpfnDlgProc);
                        FreeProcInstance(lpfnDlgProc );
                };


                /* replace current bloc */
                if (strcmp(bufrempl,"") != 0)
                        if(strcmp(selection,"") == 0)
                                (*pTpsStringInsert)(lVal, bufrempl, TRUE );
                        else (*pTpsStringReplace)(lVal, bufrempl, TRUE );

                /* restore position */
                (*pTpsRestorePosition)(lVal, iPos, TRUE );
                /*mise a jour des buffers*/
                strcpy(bufmot,"");
                /*strcpy(bufsource,""); */
                strcpy(azBuffer,"");


                break;
        }
```

```
}


GLOBAL      void    PFAR  TpsNotifyEvent(LONG lVal, WORD wEvent, LONG lParam1,
LONG lParam2 )
{

MLOCAL      BOOL fService1 = FALSE;


switch (wEvent)
{
        case PXAEVENT_APPLICATION_INIT   :
        {
                LOCAL       BYTE azItem[30];

                strcpy(azItem, azMenuItem );
                /* try to insert item into current menu */
                if ((*pTpsMenuItemInsert)("Dictionnaire-multilingue...", azItem, 1,
                        SRV_MENUID_MAINALL, SRV_MENUITEM_BEHIND) != 0 )
                        fService1 = TRUE;


        }
        break;

        case PXAEVENT_APPLICATION_EXIT  :
        break;
        case PXAEVENT_DOCUMENT_INIT             :
        break;
        case PXAEVENT_DOCUMENT_EXIT             :
        break;
        case PXAEVENT_MENU_CHANGE               :
        {
                LOCAL       BYTE azItem[30];
                strcpy(azItem, azMenuItem );
                /* test if item inserted already or try again */
                if ((fService1 == FALSE) && ((*pTpsMenuItemInsert)("Dictionnaire-
multilingue...", azItem, 1, SRV_MENUID_MAINALL, SRV_MENUITEM_BEHIND) != 0) )
                        fService1 = TRUE;

        }
        break;
        default :
        break;
}
}
```

```
GLOBAL      void    PFAR  TpsProc(WORD wSrvFunctionId, TS_MISC pSrvFunction )
{

switch (wSrvFunctionId)
{
        case PS_DEBUG_STRING_WRITE          :
                pTpsDebugStringWrite = (TS_DEBUG_STRING_WRITE)pSrvFunction;
                break;
        case PS_OK_BOX                      :
                pTpsOkBox = (TS_OK_BOX)pSrvFunction;
                break;
        case PS_OK_CANCEL_BOX               :
                pTpsOkCancelBox = (TS_OK_CANCEL_BOX)pSrvFunction;
                break;
        case PS_MENUITEM_INSERT             :
                pTpsMenuItemInsert = (TS_MENUITEM_INSERT)pSrvFunction;
                break;
        case PS_MENUITEM_DELETE             :
                pTpsMenuItemDelete = (TS_MENUITEM_DELETE)pSrvFunction;
                break;
        case PS_SAVE_POSITION               :
                pTpsSavePosition = (TS_SAVE_POSITION)pSrvFunction;
                break;
        case PS_RESTORE_POSITION            :
                pTpsRestorePosition = (TS_RESTORE_POSITION)pSrvFunction;
                break;
        case PS_MOVETO_NEXT_WORD :
                pTpsMoveToNextWord = (TS_MOVETO_NEXT_WORD)pSrvFunction;
                break;
        case PS_MOVETO_PREVIOUS_WORD        :
                pTpsMoveToPreviousWord =
                        (TS_MOVETO_PREVIOUS_WORD)pSrvFunction;
                break;
        case PS_QUERY_EXISTS_SELECTION      :
                pTpsQueryExistsSelection =
                        (TS_QUERY_EXIST_SELECTION)pSrvFunction;
                break;
        case PS_QUERY_SELECTION             :
                pTpsQuerySelection = (TS_QUERY_SELECTION)pSrvFunction;
                break;
        case PS_SELECT_WORD                 :
                pTpsSelectWord = (TS_SELECT_WORD)pSrvFunction;
                break;
        case PS_STRING_REPLACE              :
                pTpsStringReplace = (TS_STRING_REPLACE)pSrvFunction;
                break;
        case PS_STRING_INSERT               :
                pTpsStringInsert = (TS_STRING_INSERT)pSrvFunction;
```

```
            break;
        case PS_QUERY_PROGRAM_PATH                  :
            pTpsQueryProgramPath = (TS_QUERY_PROGRAM_PATH)pSrvFunction;
            break;
        default :
            break;
};

}
```